COA-Secure Obfuscation and Applications*

Ran Canetti † Suvradip Chakraborty ‡ Dakshita Khurana § Nishant Kumar ¶
Oxana Poburinnaya || Manoj Prabhakaran **

December 31, 2022

Abstract

We put forth a new paradigm for program obfuscation, where obfuscated programs are endowed with proofs of "well-formedness." In addition to asserting existence of an underlying plaintext program with an attested structure and functionality, these proofs also prevent mauling attacks, whereby an adversary surreptitiously creates an obfuscated program based on secrets which are embedded in a given obfuscated program. We call this new guarantee *Chosen Obfuscation Attack (COA) security*.

We define and construct general-purpose COA-secure Probabilistic Indistinguishability Obfuscators for circuits, assuming sub-exponential IO for circuits and one-way functions. To demonstrate the power of the new notion, we use it to realize, in the plain model:

- A new form of software watermarking, which provides significantly broader protection than current schemes and features a keyless, public verification process.
- *Completely CCA* encryption, which is a strengthening of completely non-malleable encryption.

We also show, based on the same assumptions, a generic method for enhancing any obfuscation mechanism that guarantees semantic-style hiding to one that provides COA security.

*An extended abstract of this work appears in the proceedings of Eurocrypt 2022.

†Boston University. Email: canetti@bu.edu. Supported by the DARPA SIEVE project, contract No. #HR00112020023.

‡ETH Zurich. Email: suvradip1111@gmail.com (work done while at IST Austria). Supported in part by ERC grant 724307.

§UIUC. Email: dakshita@illinois.edu. Supported in part by DARPA SIEVE project contract No. #HR00112020024, a gift from Visa Research, and a C3AI DTI award

¶Work done at UIUC.

||Email: oxanapob@bu.edu

**IIT Bombay. Email: mp@cse.iitb.ac.in. Supported by a Ramanujan Fellowship and Joint Indo-Israel Project DST/INT/ISR/P-16/2017 of Dept. of Science and Technology, India.

In loving memory of Nishant (December 2, 1994 - April 10, 2022), who was deeply passionate about cryptography.

Contents

1 Introduction

General-purpose program obfuscation (developed in [3, 12, 16, 5, 13, 10] and many other works) holds great promise for enhancing the security of software: Software can be distributed and executed without fear of exposing sensitive design secrets or keys hidden in the code. Furthermore, when executing obfuscated software, all intermediate states are guaranteed to remain hidden, even when both the hardware and the software components of the underlying platform are adversarial.

However, ubiquitous use of program obfuscation might actually have a negative effect on other security aspects of software: verifying properties of an obfuscated program may be significantly harder - it is essentially reduced to black-box testing the program. This is highly unsatisfactory, especially in situations where the source of the program is untrusted. Indeed, the very property that makes obfuscation a boon for software creators - namely the ability to hide secrets both in the code and in the functionality - is a bane for users of the software, unless those users put complete trust in the software creators and vendors.

To the best of our knowledge, the only existing general notion of obfuscation that provides the ability to verify properties of obfuscated programs is *verifiable indistinguishability obfuscation* [2]. However, that notion provides only limited hiding guarantees: indistinguishability of the obfuscated versions of two functionally equivalent programs is guaranteed only if there is a short witness of their equivalence. Furthermore, it provides no guarantees against adversaries that maul programs that were honestly generated and obfuscated.

Another concern is that the use of program obfuscation makes it harder to verify whether a given program depends on other programs in "illegitimate ways", where legitimacy (and lack thereof) relates to both structural and functional dependence between programs. For instance, obfuscation might facilitate *software plagiarism* by hiding the fact that program A runs some (potentially proprietary) program B as a subroutine, without publicly disclosing this fact. Furthermore, obfuscation may help an adversary hide the fact that a program A that they are publishing is a *mauled* version of B - i.e. that A's functionality surreptitiously depends on the functionality of B. The latter can be a concern even regardless of how B is implemented[1].

Countering mauling attacks on obfuscated programs has so far been studied only in the restricted setting of virtual black box obfuscation of point functions and related functionalities [8, 22], and is susceptible to strong impossiblity results [4]. In particular, to the best of our knowledge no viable notion of non-malleability of general-purpose obfuscation has been proposed.

Defending against program plagiarism has also been studied in the context of software watermarking [4]. However, that line of work has concentrated on detecting illicit programs that very closely preserve the functionality of the watermarked program [9] (or else preseve some cryptographic use of it [15]), and does not address more general forms of plagiarism – e.g., generating a seemingly legitimate obfuscated program that illicitly uses a plagiarized program as a subroutine, or even changing some internal parameters while preserving the overall design.

1.1 Our Contributions

We first summarize our main contributions and then elaborate on each one.

- *Definitions:* We define several variants of verifiable and non-malleable obfuscation: First we formulate a new notion of verifiable obfuscation, that provides significantly stronger hiding guarantees than [2]. Next, incorporating non-malleability, we propose a notion of *verifiable obfuscation that's secure against chosen obfuscation attacks (COA)*, and offers hiding guarantees akin to those of Probabilistic Indistinguishability Obfuscation [7]. Finally we generalize the above notion to a general *verifiability and non-malleability fortifier* that transforms any notion of obfuscation (our of a large class of common notions) into one that preserves the same level of hiding, while adding verifiability and COA security.

[1] One might expect that existing notions of obfuscation, such as indistinguishability obfuscation (IO), already defend against such mauling attacks. However, this expectation fails for programs whose code includes random keys that affect the functionality. For instance, IO does not appear to rule out the possibility that an adversary, given an obfuscated version of a puncturable pseudorandom function with a random key k, manages to generate another obfuscated program that computes the same function but with key $(k+1)$.

- *Constructions:* We construct COA obfuscation, assuming subexponentially secure iO and one-way functions. More generally, we show how to enhance (or, fortify) any one out of a class of measures of secure obfuscation to provide COA security.
- *Applications:* COA-secure obfuscation is directly applicable in situations where a user wishes to verify predicates involving the structure and functionality of an obfuscated program. In addition, we use COA-secure obfuscation to construct *software watermarking* mechanisms that provide a new and powerful notion of security. Finally, we use COA-secure obfuscation to construct a *completely CCA*-secure public-key encryption scheme, which is a new notion of security that naturally strengthens *completely non-malleable* public-key encryption [11], which in turn augments non-malleablity to consider mauling attacks against both the ciphertext *and the public key*.

1.2 Defining COA Obfuscation

The first main contribution of this work is in developing a security notion for general-purpose program obfuscation that incorporates meaningful secrecy, verifiability, and non-malleability guarantees, while still being realizable under (strong but) plausible hardness assumptions.

Several challenges face such an endeavor. First, simply requiring the ability to verify general properties of an obfuscated program, while hiding "everything else" about the program immediately runs into known impossibility results regarding one-message arguments in the plain model. In particular, standard notions such as zero-knowledge, witness hiding, or even strong witness indistingushablity are not obtainable in general. Following [18], we get around this apparent barrier by allowing the verification to be randomized, and relaxing the soundness guarantee to only require that, in each execution, the verification process either fails, or else outputs *some* program that's guaranteed to have the attested property. That is, we allow the verification process, given a purported proof, to output different programs in different executions. Furthermore, while an honest obfuscator is guaranteed that the verification process always outputs a program whose functionality equals that of the plaintext program, there may well exist (adversarially generated) proofs that cause different runs of the verification process to output programs with different functionalities. As argued within, and demonstrated by the applications, this randomized verifiability notion is still useful.

Developing an appropriate notion of non-malleability for general program obfuscation proves equally challenging. In particular, it is not a priori clear how to capture non-malleability with respect to the functionality of programs without resorting to simulation-based formalisms (as done in [8, 22]), which would in turn be subject to general impossibility akin to VBB-obfuscation [4].

We get around this difficulty by adapting the notion of CCA-secure commitments (namely, commitments that are secure against chosen commitment attacks [6]), which are in turn an extension of the notion of CCA-secure public-key encryption [25, 26], to the setting of indistinguishability obfuscation. Indeed, as there, an indistingushability-based definition which considers adversaries that have access to (potentially inefficient) oracle that conditionally inverts the cryptographic transform at hand, turns out to capture non-malleability in a strong way.

More specifically, to capture verifability we consider obfuscators O that take as input a program C along with a predicate ϕ that represents some attestation on the structure and functionality of C.

The process of executing an obfuscated program is now preceded by running a randomized verification algorithm V that, given a purported obfuscated program $\widehat{C}$, outputs either a reject symbol, or else another program $\widetilde{C}$. We then require:

Completeness: Whenever $\phi(C)$ holds, $V(O(C,\phi),\phi) = \widetilde{C}$, where $\widetilde{C}$ is functionally equivalent to C.

Soundness: For all strings $\widehat{C}$ and predicates ϕ it holds that, except for negligible probability, either $V(\widehat{C},\phi) \neq \perp$ or else $V(\widehat{C},\phi) = \widetilde{C}$ where $\widetilde{C}$ is functionally equivalent to a program C such that $\phi(C)$ holds.

Said otherwise, our notion postulates a two-step process for generating obfuscated programs: The first step is performed by the obfuscating party and generates a *semi-functional* program $\widehat{C} = O(C,\phi)$. The

second step is performed by the party intending to run the obfuscated program, and returns a *fully functional* program $\widetilde{C} = V(O(C,\phi),\phi)$ that's functionally equivalent to C. While soundness guarantees that, for any $\widehat{C}$, the random variable $\widetilde{C} = V(O(\widehat{C},\phi)$ is almost always functionally equivalent to a program C such that $\phi(C)$ holds, it is stressed that different draws from $\widetilde{C}$ might well have different functionality.

Next we proceed to formulizing the hiding requirement. We would like to require that, for "sufficiently similar" programs C_0, C_1, polytime adversaries be unable to distinguish $O(C_0,\phi)$ from $O(C_1,\phi)$, *even when given access to a de-obfuscation oracle.* That is, an adversary should be unable to guess b when given a challenge program $C^* = O(C_b,\phi)$ for $b \leftarrow \{0,1\}$, along with access to an oracle $D_{\phi,C^*}(\cdot)$ that operates as follows: $D_{\phi,C^*}(C^*) = \perp$. For $\widehat{C} \neq C^*$, D samples $\widetilde{C} = V(\widehat{C},\phi)$. and returns an arbitrary (say, the lexicographically first) program C that is functionally equivalent to $\widetilde{C}$, and such that $\phi(C)$ holds. If no such program C exists then $\perp$ is returned.

It remains to determine what makes programs C_0, C_1 "sufficiently similar". One natural option, corresponding to indistinguishability obfuscation (iO), considers any pair C_0, C_1 of functionally equivalent programs that satisfy ϕ and have the same size. However, we use a somewhat more general formulation, which turns out to be natural to work with in the context of our applications. This formulation, which corresponds to a slight simplification of the X-Ind-piO notion of obfucation of probabilistic circuits [7], proceeds as follows. We consider samplers that sample triples (C_0, C_1, z), where C_0 and C_1 are programs with input domain $\{0,1\}^\kappa$ and z is some auxiliary information. A sampler $\mathsf{Samp}(\kappa) \rightarrow (C_0, C_1, z)$ is admissible for ϕ if both C_0 and C_1 satisfy ϕ and in addition any poly(κ) adversary A, given z and oracle access to C_b, can predict b with advantage at most $\frac{1}{2} + \mathsf{negl}(2^{-\kappa})$. That is, we require:

Hiding: For any polytime adversary A and sampler Samp that is admissible for ϕ, we consider the experiment where A is given $C^* = O(C_b,\phi), z$ for $(C_0, C_1, z) \leftarrow \mathsf{Samp}(\kappa)$ and $b \leftarrow \{0,1\}$, as well access to a deobfuscation oracle $D_{\phi,C^*}(\cdot)$. Obfuscator O is hiding if A guesses b only with advantage $\mathsf{negl}(\kappa)$.

Informally, the reasoning for why COA security guarantees non-malleability is the same as in the case of CCA commitment and CCA encryption: An adversary that manages to "maul" its challenge progrom $\widehat{C}$ into a program $\widehat{C}'$ that passes verification and such that the preimages of the resulting $\mathsf{Verify}(\widehat{C})$ and $\mathsf{Verify}(\widehat{C}')$ are related in some non-trivial way, can readily use this ability to break COA security, by applying $D(\widehat{C}')$ to obtain the plaintext program that is related to the preimage of its challenge C^*. It is stressed that here the "non trivial relation" may include both structural and functional properties of the plaintext programs.

Finally, an obfuscator (O, V) is said to be COA-Secure with respect to predicate ϕ if it satisfies completeness, soundness, and hiding.

COA fortification. We also consider an alternative approach to obtaining verifiability and non-malleability, which proceeds by provding a general transformation that *fortifies* existing obfuscators so as to provide verifiability and non-malleability relative to their existing hiding guarantees. We elaborate more within.

1.3 Applications of COA obfuscation

COA-secure obfuscation is directly applicable in situations where a user wishes to make sure that it has a program that satisfies a certain set of properties — and is agnostic to whether the program has any additional properties other than those asserted.

We further demonstrate the usefulness of this notion via two applications: The first is a new notion of *program watermarking* which essentially enables tamper-detection for a large class of watermarked circuits and tampering attacks. The second is a new notion of security for encryption schemes, called *completely-CCA-secure encryption.* The new notion naturally strengthens completely non-malleable encryption of [11], namely encryption that remain secure even in the presence of an oracle that decrypts adversarially chosen ciphertexts *with respect to adversarially chosen public keys.*

A new approach to software watermarking. Existing approaches to watermarking (e.g. [19, 9, 14, 15, 20]) concentrate on a watermarking process where the watermark is embedded in a program by way of changing its *functionality* on few "hidden" inputs. Similarly, detecting whether a given program bears a watermark is determined exclusively via inspecting the functionality of the program. So far, this approach has only been succesfully used to detect "counterfeit" programs that are *close in functionality* to a watermarked program (where the notion of closeness varies somewhat in different contexts). Furthermore, it requires a key generation process where the owner of the program generates a private key that's used for watermarking and an associated public key that's used for watermark detection.

We propose an alternative approach where the process of detecting whether a given program bears a watermark is done via a combination of inspecting the functionality of the program *and verifying its structural properties.* This allows us to avoid the use of dedicated keys, and make the functional changes minimal (and obvious). More importantly, it allows for detecting a significantly broader spectrum of plagiarism and piracy of software than mere functional similarity.

On the down side, our approach mandates that legitimate users of software use software that has some specific structure - in the sense that software that lacks the structure is considered counterfeit (or, at least suspect) by default. Still, this structure is fixed, public, easy to preserve and detect, and does not restrict the functionality of programs.

Specifically, we define and construct structural watermarking. Essentially, a marking algorithms M and a testing algorithm T are a structural watermarking scheme with respect to a class $\mathcal{C}$ of programs, a distribution $\mathcal{D}$ over programs in $\mathcal{C}$, a set $\mathcal{M}$ of watermarks, and a relation R on pairs of programs, if the following conditions hold:

Completeness: First, given any program $C \in \mathcal{C}$ and a mark $m \in \mathcal{M}$, algorithm M generates a program $\widehat{C}$ that bears the watermark m, in the sense that $T(\widehat{C}) = (\widetilde{C}, m)$, where $\widetilde{C}$ is functionally equivalent to C.

Unremovability: No polytime adversary, given $\widehat{C} = M(C, m)$ where C is drawn from $\mathcal{D}$ and m is a mark of the adversary's choice, can feasibly generate a counterfeit program C^* where $T(C^*) = (\widetilde{C^*}, m')$ for $m' \neq m$, and either $\widetilde{C^*}$ is not functionally equivalent to any program in $\mathcal{C}$ or else $\widetilde{C^*}$ is functionally equivalent to a program $C^* \in \mathcal{C}$ such that $R(C, C^*)$ holds.

This notion is incomparable with aforementioned literature on watermarking, in that it considers programs that were not generated via the prescribed watermarking algorithm as 'counterfeit by default'.

Nevertheless, unlike all prior definitions, both the watermarking and the detection algorithms are fixed and keyless, which means they are inherently public and universal. This calls for a use case where software vendors watermark all software and software users are expected to check for existence of watermarks by default. In such a setting, structural watermatrking prevents a significantly larger class of "software piracy" attacks than prior work, which was all tailored for programs realizing specific cryptographic functionalities such as PRFs, digital signatures or encryption schemes.

In particular, our notion prevents any attack where the "forbidden correlation" between the counterfeit and the original program may be defined via a relation that considers both the *structure* and the *functionality* of the tampered program and the original watermarked program. It can thus be used to prevent situations where the plagiarized program has very different functionality than the watermarked one, and yet uses the watermarked program as a subroutine or otherwise incorporates in its code the watermarked program (or pieces of it) in ways that are not explicitly allowed by the prescribed relation.

It is stressed however that our notion restricts attention to situations where the legitimate execution of programs involves the derivation of the fully executable program from a given "master copy" via the prescribed derivation algorithm, and retaining the master copy for further inspection.

We then use subexponential COA obfuscation to construct watermarking schemes for any function family $\mathcal{C}$, distribution $\mathcal{D}$, and relation R where:
(a) the description of a program $C \leftarrow \mathcal{D}$ is "one way" with respect to the functionality of the program (i.e. the description is uniquely determined, yet hard to efficiently extract, given sufficiently many input-output pairs), and:

(b) R is such that whenever $R(C, C^*)$ holds, knowledge of C^* enables breaking the one-wayness of C. That is, there is an algorithm that computes the description of C, given only C^* and oracle access to C.

As a concrete example, we consider watermarking a PRF family $\mathcal{D}$ where for any two keys $k \neq k'$ the functions $C_k, C_{k'}$ are distinct, and the relation $R(C_k, C_{k'})$ holds whenever $k = f(k')$ for any injective function on keys (eg, $f(k) = k$, or $f(k) = k + 1$, etc.). Alternatively, if $\mathcal{D}$ is "key-injective," then R can hold whenever $C(x) = C'(x)$ on some known input x. Alternatively, R can hold whenever C' is of the form $D(C_k(\cdot))$, where D is a known algorithm that generates a sample from some distribution using randomness provided in its input.

Application to Completely CCA Encryption. Fischlin [11] observes that many known public-key encryption - even non-malleable ones - allow for attacks where an adversary, given a legitimately generated public key pk and ciphertext $c = Enc(pk, m)$, can generate a "mauled" pair (pk', c') where $c' = Enc(pk', m')$ for a message m' that is related to m in some adversarial way. He then proceeded to define encryption schemes that thwart such attacks, and constructed such schemes (called *completely non-malleable*) in the random oracle model. He also showed that completely non-malleable encryption is *impossible* to realize via black-box reductions to falsifiable polynomial-time hardness assumptions in the plain model (i.e., without any trusted setup assumptions).

We formulate *completely CCA (CCCA) secure encryption*, which naturally extends completely non-malleable encryption analogously to the way CCA encryption extends non-malleable encryption. That is, our notion requires that an adversary, given (pk^*, c^*) where $(pk^*, sk^*) \leftarrow Gen()$ and $c^* = Enc(pk^*, m_b)$, for adversarially chosen m_0, m_1 and $b \leftarrow \{0, 1\}$, be unable to guess b significantly better than random, even when given access to the following oracle D: Given any pair (pk, c) such that pk is not 'obviously fake', and either $pk \neq pk^*$ or $c \neq c^*$, D returns the most probable m such that $c = Enc(pk, m)$, or $\perp$ if no such m exists. (A string pk is 'obviously fake' if $Enc(pk, m) = \perp$ for any m, except for negligible probability.) It is stressed that the adversary can query D adaptively with different (possibly related) combinations of public keys and ciphertexts.

We then show that the following adaptation of the Sahai-Waters CCA secure encryption [27] is CCCA secure, in the plain model: First, we replace the plain iO obfuscator of [27] with a COA-secure obfuscator with respect to a predicate ϕ that attests for the correct structure of the obfuscated program. Next, we augment the encryption algorithm by the following "sanity check": Recall that in [27] the public key is an obfuscated program, which means that in our case the public key is a semi-functional obfuscated program $\widehat{C}$. To encrypt a message m, the encryption algorithm will run $V(\widehat{C})$ (where V is the verification step of the COA obfuscation) κ times, obtaining fully functional programs $\widetilde{C}_1, ..., \widetilde{C}_\kappa$. It will then pick a random r and compute $c_i = \widetilde{C}_i(m, r)$, $i = 1..\kappa$. If all the c_i's are equal, the final ciphertext is c_1. Else the encryption algorithm outputs $\perp$.

We note that, while our proof does proceed via black-box reductions to game-based assumptions, Fischlin's impossibility result of is evaded by the fact that the security of our COA obfuscation is proven by reduction to the *sub-exponential* security of the underlying IO and one way functions. (In a way, this demonstrates the necessity of our reliance on sub-exponential hardness to begin with.)

1.4 Constructing COA obfuscation

Our construction uses three main components. The first component is a program obfuscation scheme that guarantees polynomial slowdown, functionality preservation, and some form of hiding. (For simplicity of exposition, consider indistingushability obfuscators; however, as argued below, the construction and analysis generalize to a large class of hiding properties of the underlying obfuscation scheme.)

The second component is non-interactive distributionally indistinguishable (NIDI) proofs [18]. These are one-message protocols for a prover P_L and verifier V_L for a language $L \in NP$, whereby both the prover and the verifier are randomized, and the prover can help the verifier generate instances drawn from a given (secret) distribution D_x which is the projection on the first component of a distribution D

over pairs $(x, w) \in R_L$, while guaranteeing that: (a). Both the full distribution D and the projection on w of the sampled points remain hidden from the verifier, namely if $D_x \approx D'_x$ then $P_L(D) \approx P_L(D')$; (b). all instances that the verifier draws from the distribution are in L, namely for any purported proof π, if $V_L(\pi) = \widetilde{D}$ where $\widetilde{D} \neq \perp$, then $\widetilde{D}$ describes a distribution over $x \in L$. In [18], NIDI proofs for NP are constructed from subexponentially secure IO for circuits and one way functions.

Randomized Verifiability. As a first step towards constructing COA obfuscation, we construct IO with plain randomized verifiability. For this purpose we use an IO obfuscator O, a NIDI for NP, and a non-interactive statistically binding commitment scheme Com. Specifically, to verifiably obfuscate a circuit C^* with respect to some property ϕ, consider the NIDI (P_L, V_L) for the following language and distribution:

- The language $L = \{(\widetilde{C}, c) : \exists C, r_{obf}, r_{com}\ s.t.\ \phi(C) = 1, \tilde{C} = O(C, r_{obf}), c = Com(C, r_{com})\}$
- The distribution D^{C^*} which outputs $\perp$ if $\phi(C^*) = 0$, and otherwise draws r_{obf}, r_{com} at random and outputs $((O(C, r_{obf}), Com(C, r_{com})), (C, r_{obf}, r_{com}))$.

The obfuscator now outputs $\widehat{C} = P_L(D^{C^*})$. The obfuscation verifier runs $V_L(\widehat{C})$.

Soundness follows from the soundness of the underlying NIDI and the functionality preservation of the underlying obfuscation scheme. Plain (non-COA) hiding follows from the hiding of the commitment, the NIDI, and the underlying obfuscation. The statistical binding property of t he commitment is used to argue completeness - specifically in the definition of the language L that's the starting point of the NIDI.

Obtaining COA security. COA security is obtained via the same construction, where the commitment is upgraded to CCA-secure non-interactive commitment. CCA-secure commitments, introduced by [6], guarantee hiding even against an adversary that has access to a de-commitment oracle that returns the committed plaintext given any valid commitment string (or transcript) that's different from the challenge one. Khurana [18] constructs non-interactive CCA secure commitment schemes from NIDI arguments.

Robust NIDI. At high level, the reduction from an attacker that breaks the COA security of the obfuscation to an adversary that breaks the CCA security of the underlying commitment is straightforward. However, there is a caveat: The use of CCA-secure commitment schemes means that a stronger form of NIDI is needed to begin with: We need the NIDI to be robust against certain forms of inefficient attacks. (We borrow the term from [6] where it was used in a similar sense.) Specifically, robust NIDI w.r.t. an oracle $\mathbb{O}$ retains its indistinguishability preservation guarantee for distinguishers that have access to $\mathbb{O}$.

Our construction of a robust NIDI follows the outline in [18], except that we instantiate all primitives with those that retain their security guarantees in the presence of the oracle $\mathbb{O}$. In what follows, we outline this construction.

In a nutshell, a NIDI consists of an iO-obfuscated program that obtains as input the first message of an appropriate two-message proof system (satisfying ZK with superpolynomial simulation), and outputs a statement sampled from the input distribution, together with a proof. In [18], it was shown that the resulting system hides the distribution from which statements are sampled. Our construction of robust NIDIs modifies this template by requiring the underlying iO and ZK proof to be secure in the presence of the oracle $\mathbb{O}$. For any oracle with a finite truth table, we achieve this by assuming subexponential security of the underlying primitives, eg., by setting the iO security parameter large enough such that iO becomes secure against adversaries that store the underlying truth table.

General fortification of obfuscators. As can be seen from the above description, the construction and analysis is relatively agnostic to the specific hiding property obtained by the underlying obfuscator. In other words, the construction can be viewed as a general way to "fortify" a given obfuscator O that guarantees some hiding property X to an obfuscator O' that guarantees a "COA version" of X.

In particular, when the underlying obfuscator is an X-Ind-pIO obfuscator of probabilistic circuits [7], the resulting obfuscator is COA-secure as defined in Section 1.2. The construction is then completed by recalling that [7] construct X-Ind-pIO obfuscators for general circuits from sub-exponentially secure IO and one way functions.

2 Preliminaries

We use $x \leftarrow S$ to denote uniform sampling of x from the set S. $[n]$ is used to denote the set $\{1, 2, \dots n\}$. For $x, y \in \{0,1\}^n$, $\langle x, y \rangle$ denotes the inner product of x, y, i.e. if $x = x[1 \dots n], y = y[1 \dots n]$, $\langle x, y \rangle = \bigoplus_{i \in [n]} x_i \cdot y_i$. Functional equivalance of two circuits C_1, C_2 is denoted by $C_1 \equiv C_2$. We refer to a circuit class as $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa \in \mathbb{N}}$, where $\mathcal{C}_\kappa$ consists of a set of circuits. In addition, whenever we consider a circuit class, we assume that it has a corresponding efficient predicate to check membership in the class, i.e. for circuit class $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa \in \mathbb{N}}$, there is a corresponding efficient predicate $\phi_{\mathcal{C}}$ s.t. $\phi_{\mathcal{C}}(\kappa, C) = 1$ if $C \in \mathcal{C}_\kappa$ and 0 otherwise. For a distribution $\mathcal{D}$ on domain $\mathcal{X}$, $\mathsf{Supp}(\mathcal{D})$ denotes the support of $\mathcal{D}$ on $\mathcal{X}$. We define puncturable PRFs and key-injectivity for puncturable PRFs below:

Definition 2.1 (Puncturable PRF)**.** For sets $\{0,1\}^n$ and $\{0,1\}^m$, a puncturable PRF with key space $\mathcal{K}$ consists of a tuple of algorithms $(\mathsf{PRF.Eval},$
$\mathsf{PRF.Puncture}, \mathsf{PRF.pEval})$ that satisfy the following two conditions.

- **Functionality preserving under puncturing.** For every $x^* \in \{0,1\}^n$, every $x \in \{0,1\}^n \setminus \{x^*\}$, and all $K \in \mathcal{K}$, we have: $\mathsf{PRF.Eval}(K, x) = \mathsf{PRF.pEval}(K\{x^*\}, x)$, where $K\{x^*\} \leftarrow \mathsf{PRF.Puncture}(K, x^*)$.
- **Pseudorandomness at punctured points.** For every $x^* \in \{0,1\}^n$, every $x \in \{0,1\}^n \setminus \{x^*\}$, and any PPT adversary $\mathcal{A}$, it holds that

$$\left| \Pr\left[\mathcal{A}(K\{x^*\}, \mathsf{PRF.Eval}(K, x^*)) = 1\right] - \Pr[\mathcal{A}(K\{x^*\}, U_k) = 1] \right| = \mathsf{negl}(\kappa),$$

  where $K \leftarrow \mathcal{K}, K\{x^*\} \leftarrow \mathsf{PRF.Puncture}(K, x^*)$, and U_k is the uniform distribution over $\{0,1\}^k$.

2.1 Non-Interactive Distributionally Indistinguishable (NIDI) Arguments.

In a NIDI argument for an NP language $\mathcal{L}$, the prover algorithm $\mathcal{P}$ is given a distribution $\mathcal{D}$ for sampling member-witness pairs, and it generates a program π which can be used (by the verifier algorithm $\mathcal{V}$) to verifiably generate a member of the language $\mathcal{L}$. The hiding property of a NIDI is that if two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ are such that the members they generate are indistinguishable from each other (when the witnesses are held back), then the program π generated by the NIDI prover remains similarly indistinguishable, upto a "gap" ϵ. We formally recall the definition of this primitive from [18] below.

Definition 2.2 (Non-Interactive Distributionally-Indistinguishable (NIDI) Arguments)**.** A pair of PPT algorithms $(\mathcal{P}, \mathcal{V})$ is a non-interactive distributionally-indistinguishable (NIDI) argument for NP language $\mathcal{L}$ with associated relation $R_{\mathcal{L}}$ if there exist non-interactive algorithms $\mathcal{P}$ and $\mathcal{V}$ that satisfy:

- **Completeness:** For every $\mathsf{poly}(\kappa)$-sampleable distribution[2] $\mathcal{D} = (\mathcal{X}, \mathcal{W})$ over instance-witness pairs in $R_{\mathcal{L}}$ such that $\mathsf{Supp}(\mathcal{X}) \subseteq \mathcal{L}$,

$$\pi \in \mathsf{Supp}\left(\mathcal{P}(1^\kappa, \mathcal{D})\right) \implies \mathcal{V}(1^\kappa, \pi) \in \mathsf{Supp}(\mathcal{X}).$$

- **Soundness:** For every ensemble of polynomial-length strings $\{\pi_\kappa\}_\kappa$ there exists a negligible function μ such that

$$\Pr_{x \leftarrow \mathcal{V}(1^\kappa, \pi_\kappa)} \left[(x \neq \bot) \wedge (x \notin \mathcal{L}) \right] \leq \mu(\kappa).$$

[2]Here, we slightly abuse notation and use $\mathcal{D}$ to also denote a circuit that on input uniform randomness, outputs a sample from the distribution $\mathcal{D}$.

- **ϵ-Gap Distributional Indistinguishability:** There exists an efficient transformation T on distinguishers such that for every $\mathsf{poly}(\kappa)$-sampleable pair of distributions $\mathcal{D}_0 = (\mathcal{X}_0, \mathcal{W}_0)$ and $\mathcal{D}_1 = (\mathcal{X}_1, \mathcal{W}_1)$ over instance-witness pairs in $R_{\mathcal{L}}$ where $\mathsf{Supp}(\mathcal{X}_0) \cup \mathsf{Supp}(\mathcal{X}_1) \subseteq \mathcal{L}$, and every distinguisher D with

$$\left| \Pr[D(\mathcal{P}(1^\kappa, \mathcal{D}_0)) = 1] - \Pr[D(\mathcal{P}(1^\kappa, \mathcal{D}_1)) = 1] \right| = \nu(\kappa)$$

the distinguisher $D' = T(D)$ satisfies:

$$\left| \Pr[D'(\mathcal{X}_0) = 1] - \Pr[D'(\mathcal{X}_1) = 1] \right| \geq \epsilon(\kappa) \cdot \nu(\kappa).$$

Theorem 2.1. *[18] Assuming the existence of sub-exponentially secure one-way functions and sub-exponentially secure indistinguishability obfuscation, there exist* NIDI *arguments satisfying ϵ-gap distributional indistinguishability, for every $\epsilon(\kappa) = 2^{-o(\log^c \kappa)}$, for a constant $c > 1$.*

2.2 CCA Commitments

A chosen-commitment attack (CCA) secure commitment scheme [6] is a commitment scheme, which remains hiding for commitments even in the presence of a (computationally inefficient) "decommitment oracle" $\mathrm{CCA}.\mathsf{DeCom}$ that opens all commitments that do not match the challenge commitment. For the decommitment oracle to be well-defined, we shall require that the commitment is perfectly binding: for all r_0, r_1 and $m_0 \neq m_1$ we have that $\mathrm{CCA}.\mathsf{Com}(m_0; r_0) \neq \mathrm{CCA}.\mathsf{Com}(m_1; r_1)$.

A CCA secure commitment scheme is parameterized by a message length $M = M(\kappa)$; we shall consider the message space to be $\{0,1\}^M$, where M is polynomial. As defined below, a *non-interactive* CCA commitment scheme consists of an efficient randomized algorithm $\mathrm{CCA}.\mathsf{Com}$ (with an implicit "canonical opening"). We let $\mathrm{CCA}.\mathsf{DeCom}$ denote the function that maps an output of $\mathrm{CCA}.\mathsf{Com}$ to the message underlying it (or $\perp$ if no such message exists).

Definition 2.3. An $\epsilon(\kappa)$-secure *non-interactive CCA commitment scheme* over a message space $\{0,1\}^{M(\kappa)}$ consists of a randomized algorithm $\mathrm{CCA}.\mathsf{Com}$ and a deterministic algorithm $\mathrm{CCA}.\mathsf{DeCom}$, satisfying the following.

- **Correctness.** For all $m \in \{0,1\}^M$ and $r \in \{0,1\}^*$ we have that

  $$\mathrm{CCA}.\mathsf{DeCom}(\mathrm{CCA}.\mathsf{Com}(1^\kappa, m; r)) = m.$$

  (This implies perfect binding.)

- **Efficiency.** $\mathrm{CCA}.\mathsf{Com}$ runs in time $\mathsf{poly}(\kappa)$, while $\mathrm{CCA}.\mathsf{DeCom}$ runs in time $2^{O(\kappa)}$.

- **$\epsilon(\kappa)$-Security.** For a message $m \in \{0,1\}^M$ and a distinguisher $\mathcal{D}$, let

  $$p^{\mathrm{CCA}}_{\mathcal{D},m} = \Pr_{c \leftarrow \mathrm{CCA}.\mathsf{Com}(1^\kappa, m)}[\mathcal{D}^{\mathrm{CCA}.\mathsf{DeCom} \circ \mathsf{Filt}_c}(1^\kappa, c) = 1],$$

  where Filt_c is the identity function on all inputs except c, on which it outputs $\perp$. Then, for all polynomials s there is a negligible function ν such that, for all $m_1, m_2 \in \{0,1\}^M$ and all distinguishers $\mathcal{D}$ of size at most $s(\kappa)$,

  $$\left| p^{\mathrm{CCA}}_{\mathcal{D},m_1} - p^{\mathrm{CCA}}_{\mathcal{D},m_2} \right| \leq \epsilon(\kappa)\nu(\kappa).$$

Theorem 2.2. *[18] Assuming sub-exponentially secure indistinguishability obfuscation and either*

- *Sub-exponential (classical) hardness of DDH and sub-exponential quantum hardness of LWE (as used in [17]), or*
- *Sub-exponential time-lock puzzles based on the RSW assumption (as used in [24])*

there exist non-interactive CCA commitments satisfying Definition 2.3.

The assumptions in the aforementioned theorem can also be reduced by using time-lock puzzles based on iO and the existence of hard-to-parallelize languages.

2.3 Obfuscation

An *obfuscator* $\mathcal{O}$ is a randomized program that probabilistically maps a circuit from some family $\{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}}$ to another functionally equivalent circuit. We shall require an obfuscator to satisfy the following correctness and efficiency properties (with probability 1):

Functionality Preservation. For all $\kappa \in \mathbb{N}$ and all $C \in \mathcal{C}_\kappa$, $\mathcal{O}(1^\kappa, C) \equiv C$ (where $\equiv$ indicates that the two circuits are functionally equivalent).

Polynomial Slowdown. There exists a polynomial p such that for all $\kappa \in \mathbb{N}$ and all $C \in \mathcal{C}_\kappa$, $|\mathcal{O}(1^\kappa, C)| \leq p(|C|)$ (where $|\cdot|$ denotes the size of a circuit).

Efficient Obfuscation. $\mathcal{O}$ is a polynomial time algorithm. Generally, we shall also assume that the circuits in $\mathcal{C}_\kappa$ are of size at most polynomial in κ.

Security. For a sampler Samp and a distinguisher D, we define, for $b \in \{1, 2\}$,

$$p_{\mathcal{O},D}^{\mathsf{Samp},b} := \Pr_{\substack{(C_1,C_2,z)\leftarrow\mathsf{Samp}(1^\kappa)\\ \widetilde{C}\leftarrow\mathcal{O}(1^\kappa,C_b)}}\left[D(\widetilde{C}, z) = 1\right]$$

$$\text{and} \quad \mathsf{Adv}_{\mathcal{O},D}^{\mathsf{Samp}} := \left|p_{\mathcal{O},D}^{\mathsf{Samp},1} - p_{\mathcal{O},D}^{\mathsf{Samp},2}\right| \tag{1}$$

Then, an obfuscator $\mathcal{O}$ is said to be $(\mathcal{S}, \mathcal{D})$ secure, if for all $\mathsf{Samp} \in \mathcal{S}$ and $D \in \mathcal{D}$, $\mathsf{Adv}_{\mathcal{O},\mathcal{D}}^{\mathsf{Samp}}$ is negligible. In particular, for *indistinguishability obfuscation* (iO), $\mathcal{S}$ is the class of samplers which output (C_1, C_2, z) where $C_1 \equiv C_2$ and $z = (C_1, C_2)$, and $\mathcal{D}$ consists of all PPT distinguishers.

Following [7][3] we define a class of samplers, called admissible samplers, that only requires that it is (very) hard for a PPT adversary to distinguish between oracle access to C_1 and to C_2. Here the distinguishing probability is required to be negligible even after amplifying by a factor of 2^κ, with κ being the number of bits of inputs for the circuits.

Definition 2.4 (Admissible Samplers)**.** For any adversary $\mathcal{A}$ let

$$p_{\mathcal{A}}^{\mathsf{Samp},b,\kappa} := \Pr_{(C_1,C_2,z)\leftarrow\mathsf{Samp}(1^\kappa)}\left[\mathcal{A}^{C_b}(z) = 1\right], \quad b \in \{1, 2\}$$

$$\text{and} \quad \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Samp},\kappa} := \left|p_{\mathcal{A}}^{\mathsf{Samp},1,\kappa} - p_{\mathcal{A}}^{\mathsf{Samp},2,\kappa}\right|.$$

A sampler Samp over $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}}$ where all $C \in \mathcal{C}_\kappa$ take κ-bit inputs, is called *admissible* if there exists a negligible function μ s.t. for any non-uniform PPT adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Samp},\kappa} \leq \mu(\kappa) \cdot 2^{-\kappa}$, for all sufficiently large κ.

Definition 2.5 (pIO Obfuscators)**.** An obfuscator $\mathcal{O}$ for a circuit family $\{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}}$ is said to be pIO if it is functionality preserving, has polynomial slowdown, and in addition for any admissible sampler Samp and any non-uniform PPT adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Samp},\kappa} \leq \mu(\kappa)$ for all sufficiently large κ, where

$$p_{\mathcal{A}}^{\mathsf{Samp},b,\kappa} := \Pr_{(C_1,C_2,z)\leftarrow\mathsf{Samp}(1^\kappa)}\left[\mathcal{A}(\mathcal{O}(C_b), z) = 1\right], \quad b \in \{1, 2\}$$

$$\text{and} \quad \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Samp},\kappa} := \left|p_{\mathcal{A}}^{\mathsf{Samp},1,\kappa} - p_{\mathcal{A}}^{\mathsf{Samp},2,\kappa}\right|. \tag{2}$$

As shown in [7], assuming the existence of sub-exponentially secure iO and sub-exponentially secure puncturable PRFs, pIO schemes exist for any polynomial sized circuit family, that is secure against a class $\mathcal{D}$ of sub-exponential time distinguishers. (We note that the above defnition of pIO obfuscation corresponds to X-Ind pIO in [7].)

Next we define injective obfuscators.

[3] Admissible samplers are a special case of X-Ind sampler defined in [7], where it is parametrized by a function $X(\kappa) \leq 2^\kappa$. The definition of admissible samplers corresponds to setting $X(\kappa) = 2^\kappa$ and restricting to (deterministic) circuits taking κ-bit inputs.

Definition 2.6 (Injective Obfuscator)**.** An obfuscator $\mathcal{O}$ for a circuit family $\{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}}$ is said to be *injective* if $\forall \kappa_1, \kappa_2, C_1, C_2$

$$\mathcal{O}(1^{\kappa_1}, C_1; r_1) = \mathcal{O}(1^{\kappa_2}, C_2; r_2) \neq \perp \Rightarrow C_1 = C_2.$$

We remark that it is easy to convert any obfuscator into an injective obfuscator (without affecting its hiding properties) simply by attaching a perfectly binding commitment of the circuit to its original obfuscation.

3 Defining COA obfuscation

We define COA-secure obfuscation in Section 3.1. In Sections 3.2 and 3.2 we define the more general notion of verifiability fortification and COA fortification of existing obfuscators. Towards this, first, we will need the following definition of circuit samplers.

Definition 3.1 (ϕ-Satisfying Samplers)**.** Let $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}}$ be a circuit class and ϕ be a predicate. We say that a randomized algorithm Samp is a *ϕ-satisfying sampler over $\mathcal{C}$* if, for all large enough κ, $\mathsf{Samp}(1^\kappa)$ outputs (C_1, C_2, z) such that, with probability 1, $C_1, C_2 \in \mathcal{C}_\kappa$ and, $\phi(C_1) = \phi(C_2) = 1$.

3.1 COA-Secure Obfuscation

Definition 3.2 (Admissible ϕ-satisfying Samplers)**.** A sampler algorithm $\mathsf{Samp}(1^\kappa)$ is an *admissible ϕ-satisfying sampler over $\mathcal{C}$* if it is both admissible (according to Definition 2.4) and ϕ-satisfying (according to Definition 3.1) over $\mathcal{C}$.

Definition 3.3 (COA-Secure Obfuscation)**.** A COA-secure obfuscation for a circuit class $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}}$ w.r.t. a predicate ϕ is a pair of PPT algorithms $(c\mathcal{O}.\mathsf{Obf}, c\mathcal{O}.\mathsf{Ver})$ defined as follows[4]:

- $c\mathcal{O}.\mathsf{Obf}(1^\kappa, C, \phi) \to \widehat{C}$. This takes as input the security parameter κ, a circuit $C \in \mathcal{C}_\kappa$, a predicate ϕ, and outputs an encoding $\widehat{C}$.
- $c\mathcal{O}.\mathsf{Ver}(1^\kappa, \widehat{C}, \phi) \to \{\widetilde{C} \cup \perp\}$. This takes as input a string $\widehat{C}$, a predicate ϕ, and outputs either a circuit $\widetilde{C}$ or a reject symbol $\perp$.

Furthermore, these algorithms satisfy the following correctness, verifiability and security properties.

- **Perfect Correctness.** For every $\kappa \in \mathbb{N}$ and circuit $C \in \mathcal{C}_\kappa$ s.t. $\phi(C) = 1$, if $\widetilde{C} \leftarrow c\mathcal{O}.\mathsf{Ver}(1^\kappa, c\mathcal{O}.\mathsf{Obf}(1^\kappa, C, \phi), \phi)$, then $\widetilde{C} \equiv C$.
- **Verifiability.** For every ensemble of polynomial-length strings $\{\Pi_\kappa\}_{\kappa\in\mathbb{N}}$, there exists a negligible function $\nu(\cdot)$ such that:

$$\Pr_{\widetilde{C}\leftarrow c\mathcal{O}.\mathsf{Ver}(1^\kappa,\Pi_\kappa,\phi)}\left[\widetilde{C} \neq \perp \wedge \left(\nexists C \in \mathcal{C}_\kappa : \phi(C) = 1 \wedge \widetilde{C} \equiv C\right)\right] = \nu(\kappa).$$

- **COA Security.** Let $\mathbb{O}$ be an oracle defined as follows: $\mathbb{O}(\kappa, \widetilde{C})$ outputs the lexicographically first circuit $C \in \mathcal{C}_\kappa$ such that $\phi(C) = 1$ and C is functionally equivalent to $\widetilde{C}$.

  For any sampler algorithm Samp, and an oracle distinguisher $\mathcal{D}$, for $b \in \{1, 2\}$, let

$$q^{\mathsf{Samp},b,\kappa}_{c\mathcal{O},\mathcal{D}} := \Pr_{\substack{(C_1,C_2,z)\leftarrow\mathsf{Samp}(1^\kappa)\\ \widehat{C}\leftarrow c\mathcal{O}.\mathsf{Obf}(1^\kappa,C_b,\phi)}}\left[\mathcal{D}^{\mathbb{O}(\kappa,\cdot)\,\circ\, c\mathcal{O}.\mathsf{Ver}(1^\kappa,\cdot,\phi)\,\circ\,\mathsf{Filt}_{\widehat{C}}}(1^\kappa, \widehat{C}, z) = 1\right],$$

$$\mathsf{COAAdv}^{\mathsf{Samp},\kappa}_{c\mathcal{O},\mathcal{D}} := \left|q^{\mathsf{Samp},1,\kappa}_{c\mathcal{O},\mathcal{D}} - q^{\mathsf{Samp},2,\kappa}_{c\mathcal{O},\mathcal{D}}\right|$$

[4]Both the algorithms $c\mathcal{O}.\mathsf{Obf}$ and $c\mathcal{O}.\mathsf{Ver}$ take as input a predicate. This is to capture the uniformity of the algorithms w.r.t. ϕ.

where $\mathsf{Filt}_{\widehat{C}}$ denotes a function that behaves as the identity function on all inputs except $\widehat{C}$, on which it outputs $\perp$. (That is, given any program $\pi \neq \widehat{C}$, $\mathcal{D}$ returns the lexicographically first circuit C, with $\phi(C) = 1$, that is functionally equivalent to $\tilde{C}$, where $\tilde{C} \leftarrow c\mathcal{O}.\mathsf{Ver}(\pi, \phi)$.)

Then, there exists a T-sized transformation $\mathcal{T}$ (on distinguisher circuits) such that for any admissible ϕ-satisfying sampler Samp (according to Definition 3.2) and distinguisher $\mathcal{D}$, it holds that:

$$\mathsf{Adv}^{\mathsf{Samp},\kappa}_{\mathcal{O},\mathcal{T}(\mathcal{D})} \geq \epsilon(\kappa) \cdot \mathsf{COAAdv}^{\mathsf{Samp},\kappa}_{c\mathcal{O},\mathcal{D}}$$

where $\mathsf{Adv}^{\mathsf{Samp}}_{\mathcal{O},\mathcal{T}(\mathcal{D})}$ is as defined in (2).

While the above definition of COA security is w.r.t. admissible samplers, we can also define COA security more generally as an add-on for obfuscation schemes $\mathcal{O}$ whose security could be w.r.t. other samplers. Before presenting this notion of fortifying any obfuscation scheme with COA security, we introduce a simpler (but already useful) notion of fortifying an obfuscation scheme by adding verifiability.

3.2 Fortification of general injective Obfuscators

This section gives a variant of the notion of COA obfuscation that allows starting from any given functionality-preserving compiler on programs and adding a layer of non-malleable verifiability while preserving any existing hiding properties of the underlying compiler. This definition is incomparable to the one given in the previous section: While the present definitional approach is more general in that it applies to any hiding property of the underlying obfuscator, it is also more restrictive in that it only applies to obfuscators that are *injective* to begin with.

We split the fortification in two stages: a first stage that provides only verifiability, and a second stage that adds non-malleability (or, COA security).

Given an obfuscation scheme $\mathcal{O}$, we define its verifiability fortification w.r.t. a predicate ϕ as a pair of algorithms $(v\mathcal{O}.\mathsf{Obf}, v\mathcal{O}.\mathsf{Verify})$. The verification algorithm guarantees that, given a string Π (purportedly generated by $v\mathcal{O}.\mathsf{Obf}$), if $\widetilde{C} \leftarrow v\mathcal{O}.\mathsf{Verify}(\Pi)$ and $\widetilde{C} \neq \perp$, then there exists a circuit C which satisfies the predicate ϕ s.t. $\widetilde{C} = \mathcal{O}(C; r)$ for some randomness r.

Definition 3.4 (Verifiability Fortification for Obfuscation)**.** Let $\mathcal{O}$ be an obfuscator for a circuit class $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}}$ and ϕ be an efficiently computable predicate on circuits. An *ϵ-gap verifiability fortification of $\mathcal{O}$ w.r.t. ϕ*, is a tuple of PPT algorithms $v\mathcal{O} = (v\mathcal{O}.\mathsf{Obf}, v\mathcal{O}.\mathsf{Verify})$ that satisfy the following:

- **Correctness.** For every $\kappa \in \mathbb{N}$ and every circuit $C \in \mathcal{C}_\kappa$, such that $\phi(C) = 1$,

$$\Pr_{\widetilde{C}\leftarrow v\mathcal{O}.\mathsf{Verify}(1^\kappa, v\mathcal{O}.\mathsf{Obf}(1^\kappa, C, \phi), \phi)}[\widetilde{C} \equiv C] = 1.$$

- **Verifiability.** For every ensemble of polynomial-length strings $\{\Pi_\kappa\}_{\kappa\in\mathbb{N}}$, there exists a negligible function $\nu(\cdot)$ such that:

$$\Pr_{\widetilde{C}\leftarrow v\mathcal{O}.\mathsf{Verify}(1^\kappa, \Pi_\kappa, \phi)}\left[\widetilde{C} \neq \perp \wedge \left(\nexists(C \in \mathcal{C}_\kappa, r) : \phi(C) = 1 \wedge \widetilde{C} = \mathcal{O}(C; r)\right)\right] = \nu(\kappa).$$

- **ϵ-Gap Indistinguishability of Obfuscated Circuits.** There exists an efficient transformation $\mathcal{T}$ (on distinguisher circuits) such that for any ϕ-satisfying sampler Samp (Definition 3.1) over $\{\mathcal{C}_\kappa\}_\kappa$ and distinguisher $\mathcal{D}$,

$$\mathsf{Adv}^{\mathsf{Samp}}_{\mathcal{O},\mathcal{T}(\mathcal{D})} \geq \epsilon(\kappa) \cdot \mathsf{Adv}^{\mathsf{Samp}}_{v\mathcal{O}.\mathsf{Obf},\mathcal{D}}$$

where $\mathsf{Adv}^{\mathsf{Samp}}_{\mathcal{O}',\mathcal{D}'}$ (for $(\mathcal{O}', \mathcal{D}') = (\mathcal{O}, \mathcal{T}(\mathcal{D}))$ or $(v\mathcal{O}.\mathsf{Obf}, \mathcal{D})$) is as defined in (1).

COA Fortification. We now define COA fortification $c\mathcal{O}$ for an obfuscation scheme $\mathcal{O}$ w.r.t. a predicate ϕ. Apart from the natural correctness property, we require that $c\mathcal{O}$ satisfies verifiability w.r.t. predicate ϕ just like verifiability fortification. In addition, we want $c\mathcal{O}$ to satisfy "gap COA security", which intuitively means that any distinguisher $\mathcal{D}$ that distiguishes between $c\mathcal{O}.\mathsf{Obf}(C_1)$ and $c\mathcal{O}.\mathsf{Obf}(C_2)$ *given access to a circuit deobfuscation oracle* can be converted to a distinguisher that distinguishes $\mathcal{O}(C_1)$ from $\mathcal{O}(C_2)$ without access to any oracle. In our construction, our transformation between distinguishers is not necessarily of polynomial size in the security parameter κ – therefore, in addition to ϵ as before, we parameterize the gap security in our definition by $T = T(\kappa)$ to capture the (in)efficiency of this transformation.

Definition 3.5 (COA Fortification for Injective Obfuscators)**.** Let $\mathcal{O}$ be an injective obfuscator for a circuit class $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}}$ and ϕ be an efficiently computable predicate on circuits. A (T, ϵ)*-gap COA fortification of* $\mathcal{O}$ *w.r.t.* ϕ is a pair of PPT algorithms $c\mathcal{O} = (c\mathcal{O}.\mathsf{Obf}, c\mathcal{O}.\mathsf{Ver})$ as follows:

- $c\mathcal{O}.\mathsf{Obf}(1^\kappa, C, \phi) \to \widehat{C}$. This is a randomized algorithm that on input security parameter κ, a circuit $C \in \mathcal{C}_\kappa$, and a predicate ϕ, outputs an encoding $\widehat{C}$.
- $c\mathcal{O}.\mathsf{Ver}(1^\kappa, \widehat{C}, \phi) \to \{\widetilde{C} \cup \bot\}$. This is a randomized algorithm that on input security parameter κ, a string $\widehat{C}$, and a predicate ϕ, outputs either a circuit $\widetilde{C}$ or a reject symbol $\bot$.

These algorithms satisfy the following correctness and security properties.

- **Perfect Correctness.** For every $\kappa \in \mathbb{N}$ and every circuit $C \in \mathcal{C}_\kappa$ such that $\phi(C) = 1$,
$$\Pr_{\widetilde{C}\leftarrow c\mathcal{O}.\mathsf{Ver}(1^\kappa, c\mathcal{O}.\mathsf{Obf}(1^\kappa, C, \phi), \phi)} [\exists\, r \text{ s.t. } \widetilde{C} = \mathcal{O}(1^\kappa, C; r)] = 1$$
- **Verifiability.** For every ensemble of polynomial-length strings $\{\Pi_\kappa\}_{\kappa\in\mathbb{N}}$, there exists a negligible function $\nu(\cdot)$ such that:
$$\Pr_{\widetilde{C}\leftarrow c\mathcal{O}.\mathsf{Ver}(1^\kappa, \Pi_\kappa, \phi)} \left[\widetilde{C} \neq \bot \wedge \left(\nexists (C \in \mathcal{C}_\kappa, r) : \phi(C) = 1 \wedge \widetilde{C} = \mathcal{O}(1^\kappa, C; r)\right)\right] = \nu(\kappa).$$
- (T, ϵ)**-Gap Security.** Let $\mathcal{O}^{-1}(\widetilde{C}) = \begin{cases} C & \text{if } \exists\, (C \in \mathcal{C}_\kappa, r) \text{ s.t. } \widetilde{C} = \mathcal{O}(1^\kappa, C; r) \\ \bot & \text{otherwise.} \end{cases}$ (well-defined since $\mathcal{O}$ is injective). For any ϕ-satisfying sampler Samp (see Definition 3.1), and an oracle circuit $\mathcal{D}$, for $b \in \{1, 2\}$, let
$$q^{\mathsf{Samp},b}_{c\mathcal{O},\mathcal{D}} := \Pr_{\substack{(C_1,C_2,z)\leftarrow\mathsf{Samp}(1^\kappa) \\ \widetilde{C}\leftarrow c\mathcal{O}.\mathsf{Obf}(1^\kappa, C_b, \phi)}} \left[\mathcal{D}^{\mathcal{O}^{-1}\circ c\mathcal{O}.\mathsf{Ver}(1^\kappa,\cdot,\phi)\circ \mathsf{Filt}_{\widehat{C}}}(1^\kappa, \widehat{C}, z) = 1\right]$$
$$\mathsf{COAAdv}^{\mathsf{Samp}}_{c\mathcal{O},\mathcal{D}} := \left|q^{\mathsf{Samp},1}_{c\mathcal{O},\mathcal{D}} - q^{\mathsf{Samp},2}_{c\mathcal{O},\mathcal{D}}\right|$$
where $\mathsf{Filt}_{\widehat{C}}$ denotes a function that behaves as the identity function on all inputs except $\widehat{C}$, on which it outputs $\bot$.

  Then, there exists a T-sized transformation $\mathcal{T}$ (on distinguisher circuits) such that for any admissible sampler Samp over $\{\mathcal{C}_\kappa\}_\kappa$ and distinguisher $\mathcal{D}$,
$$\mathsf{Adv}^{\mathsf{Samp}}_{\mathcal{O},\mathcal{T}(\mathcal{D})} \geq \epsilon(\kappa) \cdot \mathsf{COAAdv}^{\mathsf{Samp}}_{c\mathcal{O},\mathcal{D}}$$
  where $\mathsf{Adv}^{\mathsf{Samp}}_{\mathcal{O},\mathcal{T}(\mathcal{D})}$ is as defined in (1).

Remark 3.1. One could consider a (possibly) stronger definition that allows the sampler Samp used in defining $\mathsf{COAAdv}^{\mathsf{Samp}}_{c\mathcal{O},\mathcal{D}}$ to also make de-obfuscation queries. We note that for worst-case indistinguishability notions for $\mathcal{O}$ (like iO), this does not make any difference, as the (non-uniform) sampler can output the optimal pair of circuits.

Remark 3.2. We remark that for any $T = T(\kappa) \geq \mathsf{poly}(\kappa)$, any $\epsilon = \epsilon(\kappa) \leq \mathsf{negl}(\kappa)$, (T, ϵ)-gap COA fortification for any injective (T, ϵ)-secure piO implies COA-secure obfuscation according to Definition 3.3. Here (T, ϵ)-security indicates that the advantage of any $\mathsf{poly}(T)$-sized adversary in the piO security game is at most $\mathsf{negl}(\epsilon)$.

4 Robust NIDI

Robust NIDI arguments w.r.t. an oracle $\mathbb{O}$ are an extension of NIDI arguments (Definition 2.2), whereby the gap distributional indistinguishability requirement of NIDI is further strengthened to hold even if the distinguisher has access to the oracle $\mathbb{O}$. (The completeness and soundness guarantees remain unchanged.) In other words, any distinguisher $\mathcal{D}^{\mathbb{O}}$, distinguishing the proofs generated by prover $\mathcal{P}$ on input the distributions on instance-witness pairs - $\mathcal{D}_0 = (\mathcal{X}_0, \mathcal{W}_0)$ or $\mathcal{D}_1 = (\mathcal{X}_1, \mathcal{W}_1)$, can be converted to an efficient distinguisher $T(D)^{\mathbb{O}}$ which distinguishes the underlying instances $\mathcal{X}_0$ or $\mathcal{X}_1$ upto a "gap" ϵ. We formally define the same below.

Definition 4.1 (Robust NIDI Arguments)**.** Let $\mathcal{L}$ be an NP language with an associated relation $R_{\mathcal{L}}$, and $\mathbb{O}$ be an arbitrary oracle. A NIDI argument for $\mathcal{L}$, $(\mathcal{P}, \mathcal{V})$ is said to be *robust w.r.t.* $\mathbb{O}$ if it satisfies the following:

- **ϵ-Gap Robust Distributional Indistinguishability:** There exists an efficient transformation T on distinguishers such that for every $\mathsf{poly}(\kappa)$-sampleable pair of distributions $\mathcal{D}_0 = (\mathcal{X}_0, \mathcal{W}_0)$ and $\mathcal{D}_1 = (\mathcal{X}_1, \mathcal{W}_1)$ over instance-witness pairs in $R_{\mathcal{L}}$ where $\mathsf{Supp}(\mathcal{X}_0) \cup \mathsf{Supp}(\mathcal{X}_1) \subseteq \mathcal{L}$, and every distinguisher D with
$$\left| \Pr[D^{\mathbb{O}}(\mathcal{P}(1^\kappa, \mathcal{D}_0)) = 1] - \Pr[D^{\mathbb{O}}(\mathcal{P}(1^\kappa, \mathcal{D}_1)) = 1] \right| = \nu(\kappa)$$
the distinguisher $\widehat{D} = T(D)$ satisfies:
$$\left| \Pr[\widehat{D}^{\mathbb{O}}(\mathcal{X}_0) = 1] - \Pr[\widehat{D}^{\mathbb{O}}(\mathcal{X}_1) = 1] \right| \geq \epsilon(\kappa) \cdot \nu(\kappa).$$
The completeness and soundness guarantees are the same as that of a NIDI.

We construct robust NIDI arguments for any finite[5] oracle $\mathbb{O} = \{\mathbb{O}_\kappa\}_{\kappa \in \mathbb{N}}$ by modifying the construction in [18] to ensure that all the underlying primitives remain secure in the presence of oracle $\mathbb{O}$. Our approach to achieve this is to rely on complexity leveraging, although it may be possible to leverage other axes of hardness in order to instantiate the underlying primitives, with those that remain secure in the presence of $\mathbb{O}$.

Construction 4.1. *Let $\epsilon > 0$ be an arbitrary small constant s.t. $\epsilon < \delta$ and:*

- *There exists a one-way function f with an* efficiently recognizable range*, i.e., given y there is an efficient (PPT) algorithm to check whether there exists a value x such that $f(x) = y$. Note that permutations have this property, because every y is in the range of the permutation.*
- *There exists a* perfectly correct, sub-exponentially secure *public-key encryption scheme with key generation, encryption and decryption algorithms* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *that for large enough security parameter k, satisfy the following - the advantage of every* $\mathsf{poly}(2^{k^\epsilon})$*-size adversary in the* $\mathsf{IND\text{-}CPA}$ *game is* $\mathsf{negl}(2^{k^\epsilon})$.
- *There exists a* sub-exponentially secure *indistinguishability obfuscation scheme* $(\mathsf{iO.Obf}, \mathsf{iO.Eval})$*, that for large enough security parameter k, satisfies the following - the advantage of every* $\mathsf{poly}(2^{k^\epsilon})$*-size adversary in the obfuscation security game is* $\mathsf{negl}(2^{k^\epsilon})$.

[5] By 'finite', we mean that there exists a constant $c > 1$ s.t. for large enough κ the oracle $\mathbb{O}_\kappa$ can be represented as a truth-table of size at most 2^{κ^c}.

- *There exists a* sub-exponentially secure *puncturable* PRF *that for large enough security parameter k, satisfies the following - the advantage of every* $\mathsf{poly}(2^{k^\epsilon})$*-size adversary in the puncturable PRF indistinguishability game is* $\mathsf{negl}(2^{k^\epsilon})$.
- *There exist* sub-exponentially secure *NIWIs that for large enough security parameter k, satisfies the following - the advantage of every* $\mathsf{poly}(2^{k^\epsilon})$*-size adversary in the NIWI indistinguishability game is* $\mathsf{negl}(2^{k^\epsilon})$.

We construct the required robust non-interactive distributionally-indistinguishable (NIDI) argument below, where letting $\mathcal{R}_\mathcal{L}$ denote the relation corresponding to NP language $\mathcal{L}$ we define

$$\mathcal{L}_{\mathsf{NIWI}} = \left\{(pk, d, c, y) : \exists (r, s, sk) \text{ s.t. } \left((d, r) \in \mathcal{R}_\mathcal{L}\right) \bigvee \left((pk, sk) \leftarrow \mathsf{KeyGen}(s) \wedge y = f(\mathsf{Dec}_{sk}(c))\right)\right\}$$

Let $c > 1$ be a constant s.t. for large enough κ the oracle $\mathbb{O}_\kappa$ can be represented as a truth-table of size at most 2^{κ^c}.

- *The prove algorithm $\mathcal{P}(1^\kappa, \mathcal{D}, \mathcal{L})$ does the following:*
  - *Set $k = \kappa^{\frac{c}{\epsilon}}$.*
  - *Sample $s \leftarrow \{0,1\}^k$ and set $(pk, sk) \leftarrow \mathsf{KeyGen}(1^k, s)$.*
  - *Sample $K \leftarrow \{0,1\}^k, R \leftarrow \{0,1\}^k$.*
  - *Generate program $P_{pk,K,\mathcal{D},\mathcal{L}}$ defined in Figure 1.*
  - *Compute $\widetilde{P} = \mathsf{iO.Obf}(1^k, P_{pk,K,\mathcal{D},\mathcal{L}}; R)$.*
  - *Output $(pk, \widetilde{P})$.*

Hardwired: Public key pk, Puncturable PRF Key K, Distribution $\mathcal{D}$, Language $\mathcal{L}$.

Input: Query $y \in \{0,1\}^\kappa$.

1. If $y \notin \mathsf{Range}(f)$, output $\perp$. Otherwise, continue.
2. Set $(r_1, r_2, r_3) = \mathsf{PRF}(K, y)$.
3. Set $d = \mathcal{D}(r_1)$.
4. Set $c = \mathsf{Enc}_{pk}(0^\kappa; r_2)$.
5. Set $x = (pk, d, c, y)$, $w = (r_1, 0^{2k})$. Then compute $e = \mathsf{NIWI}.\mathcal{P}(1^k, x, w, \mathcal{L}_{\mathsf{NIWI}}; r_3)$.
6. Output (x, e).

Figure 1: **Program** $P_{pk,K,\mathcal{D},\mathcal{L}}$.

- *The verify algorithm $\mathcal{V}(1^\kappa, \pi, \mathcal{L})$ on input a proof $\pi = (pk, \widetilde{P})$ does the following:*
  - *Sample $v \leftarrow \{0,1\}^\kappa$ and set $y = f(v)$.*
  - *Compute* $\mathsf{out} = \mathsf{iO.Eval}(\widetilde{P}, y)$. *Parse* $\mathsf{out} = (x, e)$ *and parse $x = (pk, d, c, y)$.*
  - *If* $\mathsf{NIWI}.\mathcal{V}(1^k, x, e, \mathcal{L}_{\mathsf{NIWI}})$ *rejects, output $\perp$ and stop.*
  - *Else output d.*

Theorem 4.1. *Fix any finite oracle $\mathbb{O} = \{\mathbb{O}_\kappa\}_{\kappa \in \mathbb{N}}$. Then the above construction satisfies Definition 4.1.*

The rest of this section is dedicated to proving Theorem 4.1. Many parts of what follows closely resemble the NIDI construction in [18], with some changes to achieve security against an oracle-aided adversary.

Lemma 4.1. *Construction 4.1 satisfies completeness according to Definition 4.1.*

Proof. The proof follows by observing that due to perfect correctness of iO, $\mathcal{V}(\pi, \mathcal{L})$ for $\pi = (pk, \widetilde{P})$ obtains (x, e) from $\widetilde{P}$, where $x = (pk, d, c, y)$. By perfect correctness of NIWI, $\mathcal{V}$ will output d with probability 1. Recall that $d = \mathcal{D}(r_1)$ by construction, and therefore $d \in \mathsf{Supp}(\mathcal{D})$. □

Lemma 4.2. *Construction 4.1 satisfies soundness according to Definition 4.1.*

Proof. Suppose there exists a $\mathsf{poly}(\kappa)$-sized (non-uniform) prover $\mathcal{P}^*$ that outputs string $\pi^* \in \{0,1\}^*$, and suppose there exists a polynomial $p(\cdot)$ such that:

$$\Pr\left[\left(\mathcal{V}(1^\kappa, \pi, \mathcal{L}) \neq \bot\right) \wedge \left(\mathcal{V}(1^\kappa, \pi, \mathcal{L}) \notin \mathcal{L}\right) \middle| \pi \leftarrow \mathcal{P}^*\right] \geq \frac{1}{p(\kappa)}$$

Then, we construct a non-uniform adversary $\mathcal{A}^*$ that contradicts one-wayness of the function f. $\mathcal{A}^*$ obtains non-uniformly a purported NIDI proof π^* for language $\mathcal{L}$, where $\pi^* = (pk^*, \widetilde{P})$, and also non-uniformly obtains the secret key sk^* for pk^* (if one exists). Otherwise, it sets sk^* to 0^k.

Next, $\mathcal{A}^*$ obtains a string y and does the following to compute $f^{-1}(y)$:

- Compute $\mathsf{out} = \mathsf{iO.Eval}(\widetilde{P}, y)$. Parse $\mathsf{out} = (x^*, e^*)$ and parse $x^* = (pk^*, d^*, c^*, y)$.
- Output $z = \mathsf{Dec}_{sk^*}(c^*)$.

Now by assumption,

$$\Pr\left[\left(\mathcal{V}(1^\kappa, \pi, \mathcal{L}) \neq \bot\right) \wedge \left(\mathcal{V}(1^\kappa, \pi, \mathcal{L}) \notin \mathcal{L}\right) \middle| \pi \leftarrow \mathcal{P}^*\right] \geq \frac{1}{p(\kappa)}$$

which by our construction implies that

$$\Pr\left[\left(\mathsf{NIWI}.\mathcal{V}(x^*, e^*, \mathcal{L}_{\mathsf{NIWI}}) \text{ accepts }\right) \wedge \left(\mathcal{V}(1^\kappa, \pi, \mathcal{L}) \notin \mathcal{L}\right) \middle| \pi \leftarrow \mathcal{P}^*\right] \geq \frac{1}{p(\kappa)} \tag{3}$$

By (perfect) soundness of the NIWI, $\mathsf{NIWI}.\mathcal{V}(x^*, e^*, \mathcal{L}_{\mathsf{NIWI}})$ accepts iff $x^* \in \mathcal{L}_{\mathsf{NIWI}}$, or equivalently for $x^* = (pk^*, d^*, c^*, y^*)$,

$$\exists (r^*, s^*, sk^*) \text{ s.t. } \left((d^*, r^*) \in \mathcal{R}_{\mathcal{L}}\right) \bigvee \left((pk^*, sk^*) \leftarrow \mathsf{KeyGen}(s^*) \wedge y = f(\mathsf{Dec}_{sk^*}(c^*))\right) \tag{4}$$

Combining equations (3) and (4),

$$\Pr\left[\left(\exists (r^*, s^*, sk^*) \text{ s.t. } \left((d^*, r^*) \in \mathcal{R}_{\mathcal{L}}\right) \bigvee \left((pk^*, sk^*) \leftarrow \mathsf{KeyGen}(s^*) \wedge y = f(\mathsf{Dec}_{sk^*}(c^*))\right)\right) \bigwedge \left(\mathcal{V}(1^\kappa, \pi, \mathcal{L}) \notin \mathcal{L}\right)\right] \geq \frac{1}{p(\kappa)}$$

By noting that $d^* \notin \mathcal{L}$ implies that there does not exist r^* such that $\left((d^*, r^*) \in \mathcal{R}_{\mathcal{L}}\right)$, we have:

$$\Pr\left[\left(\exists (r^*, s^*, sk^*) \text{ s. t. } (pk^*, sk^*) \leftarrow \mathsf{KeyGen}(s^*) \wedge y = f(\mathsf{Dec}_{sk^*}(c^*))\right) \bigwedge \left(d^* \notin \mathcal{L}\right)\right] \geq \frac{1}{p(\kappa)}$$

which in particular implies that for $(pk^*, \widetilde{P})$ and corresponding sk^* set non-uniformly by $\mathcal{A}$,

$$\Pr\left[\left(\exists(r^*, s^*, sk^*) \text{ s. t. } (pk^*, sk^*) \leftarrow \mathsf{KeyGen}(s^*) \wedge y = f(\mathsf{Dec}_{sk^*}(c^*))\right)\right] \geq \frac{1}{p(\kappa)}$$

which implies that with probability at least $\frac{1}{p(\kappa)}$, the output of $\mathcal{A}^*(y)$, which is $z = \mathsf{Dec}_{sk^*}(c^*)$ as discussed above, is such that $f(z) = y$.

Therefore $\mathcal{A}^*(y)$ runs in time $\mathsf{poly}(k) = \mathsf{poly}(\kappa^{c/\epsilon}) = \mathsf{poly}(\kappa)$ and contradicts one-wayness of the function f, as desired. This completes the proof. □

Lemma 4.3. *Construction 4.1 satisfies robust distributional indistinguishability for 2^{κ^δ}-hard distributions according to Definition 4.1*

Proof. We will now define a sequence of $(2^\kappa + 1)$ hybrid distributions, $(\mathsf{Hybrid}_0, \mathsf{Hybrid}_1, \ldots, \mathsf{Hybrid}_{2^\kappa})$ such that $\mathsf{Hybrid}_0 \equiv \mathcal{P}(1^\kappa, \mathcal{D}_0, \mathcal{L})$ and $\mathsf{Hybrid}_{2^\kappa} \equiv \mathcal{P}(1^\kappa, \mathcal{D}_1, \mathcal{L})$.

For each $i \in [0, 2^\kappa]$, the distribution Hybrid_i depends on $(1^\kappa, \mathcal{D}_0, \mathcal{D}_1, \mathcal{L}, c, \epsilon)$ and is defined as follows:

- Set $k = \kappa^{\frac{c}{\epsilon}}$.
- Sample $s \leftarrow \{0,1\}^k$ and set $(pk, sk) \leftarrow \mathsf{KeyGen}(1^k, s)$.
- Sample $K \leftarrow \{0,1\}^k, R \leftarrow \{0,1\}^k$.
- Generate program $P^i_{pk,K,\mathcal{D},\mathcal{L}}$ defined in Figure 2.
- Compute $\widetilde{P} = \mathsf{iO.Obf}(1^k, P^i_{pk,K,\mathcal{D},\mathcal{L}}; R)$.
- Output $(pk, \widetilde{P})$.

Hardwired: Public key pk, Index $i \in [0, 2^\kappa]$, Puncturable PRF Key K, Distributions $(\mathcal{D}_0, \mathcal{D}_1)$, Language $\mathcal{L}$.

Input: Query $y \in \{0,1\}^\kappa$.

1. If $y \notin \mathsf{Range}(f)$, output $\perp$. Otherwise, continue.
2. Set $(r_1, r_2, r_3) = \mathsf{PRF}(K, y)$.
3. If $y < i$, set $d = \mathcal{D}_1(r_1)$.
4. If $y \geq i$, set $d = \mathcal{D}_0(r_1)$.
5. Set $c = \mathsf{Enc}_{pk}(0^\kappa; r_2)$.
6. Set $x = (pk, d, c, y)$, $w = (r_1, 0^{2k})$. Then compute $e = \mathsf{NIWI}.\mathcal{P}(1^k, x, w, \mathcal{L}_{\mathsf{NIWI}}; r_3)$.
7. Output (x, e).

Figure 2: **Program** $P^i_{pk,K,\mathcal{D},\mathcal{L}}$.

We now prove that for every $i \in [1, 2^\kappa]$ and every (non-uniform) $\mathsf{poly}(2^{\kappa^c})$-sized distinguisher $\mathcal{A}$ there exists a negligible function $\mu(\cdot)$ such that:

$$\left|\Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1}) = 1] - \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_i) = 1]\right| = \mu(2^{k^\epsilon})$$

To prove this claim, we consider a sequence of additional sub-hybrids between Hybrid_{i-1} and Hybrid_i called $(\mathsf{Hybrid}_{i-1,1}, \ldots, \mathsf{Hybrid}_{i-1,10})$ where:

$$\mathsf{Hybrid}_{i-1,1} \equiv \mathsf{Hybrid}_{i-1} \text{ and } \mathsf{Hybrid}_{i-1,10} \equiv \mathsf{Hybrid}_i$$

- $\mathsf{Hybrid}_{i-1,1} \equiv \mathsf{Hybrid}_{i-1}$.
- $\mathsf{Hybrid}_{i-1,2}$ depends on $(1^\kappa, \mathcal{D}_0, \mathcal{D}_1, \mathcal{L}, c, \epsilon)$ and is defined as follows: (we underline the difference between $\mathsf{Hybrid}_{i-1,1}$ and $\mathsf{Hybrid}_{i-1,2}$)
  - Set $k = \kappa^{\frac{c}{\epsilon}}$.
  - Sample $s \leftarrow \{0,1\}^k$ and set $(pk, sk) \leftarrow \mathsf{KeyGen}(s)$.
  - Sample $K \leftarrow \{0,1\}^k, R \leftarrow \{0,1\}^k$.
  - Set $(r_1^*, r_2^*, r_3^*) = \mathsf{PRF}(K, i)$.
  - Set $d^* = \mathcal{D}_1(r_1^*)$ and $c^* = \mathsf{Enc}_{pk}(0^\kappa; r_2^*)$.
  - Set $x^* = (pk, d^*, c^*, i)$, $w^* = (r_1^*, 0^{2k})$. Then compute $e^* = \mathsf{NIWI}.\mathcal{P}(1^k, x^*, w^*, \mathcal{L}_{\mathsf{NIWI}}; r_3^*)$.
  - Generate program $\widetilde{P}^{i-1}_{pk,K,\mathcal{D},\mathcal{L},(x^*,e^*)}$ defined in Figure 3.
  - Compute $\widetilde{P} = \mathsf{iO}.\mathsf{Obf}(\widetilde{P}^{i-1}_{pk,K,\mathcal{D},\mathcal{L},(x^*,e^*)}; R)$.
  - Output $(pk, \widetilde{P})$.

Hardwired: Public key pk, Index $i \in [2^\kappa]$, Punctured PRF Key $K\{i\}$, (x^*, e^*), Distributions $(\mathcal{D}_0, \mathcal{D}_1)$, Language $\mathcal{L}$.

Input: Query $y \in \{0,1\}^\kappa$.

1. If $y \notin \mathsf{Range}(f)$, output $\perp$. Otherwise, continue.
2. If $y = i$, output (x^*, e^*) and stop.
3. Compute $(r_1, r_2, r_3) = \mathsf{PRF}(K, y)$.
4. If $y \leq (i-1)$, set $d = \mathcal{D}_1(r_1)$.
5. If $y \geq i$, set $d = \mathcal{D}_0(r_1)$.
6. Set $c = \mathsf{Enc}_{pk}(0^\kappa; r_2)$.
7. Set $x = (pk, d, c, y)$, $w = (r_1, 0^{\kappa+k})$. Then compute $e = \mathsf{NIWI}.\mathcal{P}(1^k, x, w, \mathcal{L}_{\mathsf{NIWI}}; r_3)$.
8. Output (x, e).

Figure 3: **Program** $\widetilde{P}^{i-1}_{pk,K,\mathcal{D},\mathcal{L},(x^*,e^*)}$.

Claim 4.1. *For every (non-uniform)* $\mathsf{poly}(2^{\kappa^c})$*-sized distinguisher* $\mathcal{A}$ *there exists a negligible function* $\mu(\cdot)$ *such that:*

$$\left| \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,1}) = 1] - \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,2}) = 1] \right| = \mu(2^{k^\epsilon})$$

Proof. Note that the programs $P^{i-1}_{pk,K,\mathcal{D},\mathcal{L}}$ and $\widetilde{P}^{i-1}_{pk,K,\mathcal{D},\mathcal{L},(x^*,e^*)}$ have identical functionality, and the oracle $\mathbb{O}$ can be implemented by a circuit of size 2^{κ^c}. Therefore, by sub-exponential security of indistinguishability obfuscation (with security parameter k), for every $i \in [1, 2^\kappa]$ and every (non-uniform) $\mathsf{poly}(2^{k^\epsilon}) = \mathsf{poly}(2^{\kappa^c})$-sized distinguisher $\mathcal{A}$ there exists a negligible function $\mu_1(\cdot)$ such that:

$$\left| \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,1}) = 1] - \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,2}) = 1] \right| \leq \mu_1(2^{k^\epsilon})$$

□

- $\mathsf{Hybrid}_{i-1,3}$ is identical to $\mathsf{Hybrid}_{i-1,2}$ except that $(r_1^*, r_2^*, r_3^*) \leftarrow \{0,1\}^{3k}$.

  Claim 4.2. *For every (non-uniform) $\mathsf{poly}(2^{\kappa^c})$-sized distinguisher $\mathcal{A}$ there exists a negligible function $\mu(\cdot)$ such that:*

  $$\left| \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,2}) = 1] - \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,3}) = 1] \right| = \mu(2^{k^\epsilon})$$

  Proof. Note that the oracle $\mathbb{O}$ can be implemented by a circuit of size $2^{\kappa^c} = 2^{k^\epsilon}$. Then by sub-exponential security of the puncturable PRF (with security parameter k), we have directly that for every $i \in [1, 2^\kappa]$ and every (non-uniform) $\mathsf{poly}(2^{k^\epsilon}) = \mathsf{poly}(2^{\kappa^c})$-sized distinguisher $\mathcal{A}$ there exists a negligible function $\mu_2(\cdot)$ such that:

  $$\left| \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,2}) = 1] - \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,3}) = 1] \right| \leq \mu_2(2^{k^\epsilon})$$

  □

- $\mathsf{Hybrid}_{i-1,4}$ is defined as follows: (we underline the difference between $\mathsf{Hybrid}_{i-1,3}$ and $\mathsf{Hybrid}_{i-1,4}$)
  - Set $k = \kappa^{\frac{1}{\epsilon}}$.
  - Sample $s \leftarrow \{0,1\}^k$ and set $(pk, sk) \leftarrow \mathsf{KeyGen}(s)$.
  - Sample $K \leftarrow \{0,1\}^k, R \leftarrow \{0,1\}^k$.
  - Set $(r_1^*, r_2^*, r_3^*) \leftarrow \{0,1\}^{2\kappa+k}$.
  - Compute (in time upto 2^κ) value s_1^* such that $f(s_1^*) = i$.
  - Set $d^* = \mathcal{D}_1(r_1^*)$ and $c^* = \mathsf{Enc}_{pk}(s_1^*; r_2^*)$.
  - Set $x^* = (pk, d^*, c^*, i)$, $w^* = (r_1^*, 0^{\kappa+k})$. Then compute $e^* = \mathsf{NIWI}.\mathcal{P}(1^k, x^*, w^*, \mathcal{L}_{\mathsf{NIWI}}; r_3^*)$.
  - Generate program $\widetilde{P}^{i-1}_{pk,K,\mathcal{D},\mathcal{L},(x^*,e^*)}$ defined in Figure 3.
  - Compute $\widetilde{P} = \mathsf{iO}.\mathsf{Obf}(\widetilde{P}^{i-1}_{pk,K,\mathcal{D},\mathcal{L},(x^*,e^*)}; R)$.
  - Output $(pk, \widetilde{P})$.

  Claim 4.3. *For every (non-uniform) $\mathsf{poly}(2^{\kappa^c})$-sized distinguisher $\mathcal{A}$ there exists a negligible function $\mu(\cdot)$ such that:*

  $$\left| \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,3}) = 1] - \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,4}) = 1] \right| = \mu(2^{k^\epsilon})$$

  Proof. We will prove this claim based on the sub-exponential IND-CPA security of the encryption scheme against non-uniform adversaries. Towards a contradiction, suppose there exists a $\mathsf{poly}(2^{\kappa^c}) = \mathsf{poly}(2^{k^\epsilon})$ sized distinguisher $\mathcal{A}$ and a polynomial $p(\cdot)$ such that

  $$\left| \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,3}) = 1] - \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,4}) = 1] \right| \geq \frac{1}{p(2^{k^\epsilon})}$$

Then there exists a non-uniform $\mathsf{poly}(2^{k^\epsilon})$ sized adversary $\mathcal{A}'$ that fixes $s_1^* = f^{-1}(i)$, then begins the experiment. It obtains c^* from the CPA challenger as either $\mathsf{Enc}_{pk}(0^\kappa; r_2^*)$ or $\mathsf{Enc}_{pk}(s_1^*; r_2^*)$ It completes the rest of the experiment according to $\mathsf{Hybrid}_{i-1,3}$ except setting c^* according to the ciphertext obtained from the external challenger. It implements the oracle $\mathbb{O}$ for $\mathcal{A}$ (this requires size 2^{κ^c}). It then mirrors the output of $\mathcal{A}$ given the resulting distribution, which implies that

$$\left| \Pr[\mathcal{A}'(\mathsf{Enc}_{pk}(0^\kappa; r_2^*)) = 1] - \Pr[\mathcal{A}'(\mathsf{Enc}_{pk}(s_1^*; r_2^*)) = 1] \right| \geq \frac{1}{p(2^{k^\epsilon})}$$

which gives a contradiction, as desired. Therefore, by sub-exponential IND-CPA security of the encryption scheme (with security parameter k), for every $i \in [1, 2^\kappa]$ and every (non-uniform) $\mathsf{poly}(2^{\kappa^c})$-sized distinguisher $\mathcal{A}$ there exists a negligible function $\mu_3(\cdot)$ such that:

$$\left| \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,3}) = 1] - \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,4}) = 1] \right| \leq \mu_3(2^{k^\epsilon})$$

□

- $\mathsf{Hybrid}_{i-1,5}$ is defined as follows: (we underline the difference between $\mathsf{Hybrid}_{i-1,4}$ and $\mathsf{Hybrid}_{i-1,5}$)
  - Set $k = \kappa^{\frac{1}{\epsilon}}$.
  - Sample $s \leftarrow \{0,1\}^k$ and set $(pk, sk) \leftarrow \mathsf{KeyGen}(s)$.
  - Sample $K \leftarrow \{0,1\}^k, R \leftarrow \{0,1\}^k$.
  - Set $(r_1^*, r_2^*, r_3^*) \leftarrow \{0,1\}^{2\kappa+k}$.
  - Compute (in time upto 2^κ) value s_1^* such that $f(s_1^*) = i$.
  - Set $d^* = \mathcal{D}_1(r_1^*)$ and $c^* = \mathsf{Enc}_{pk}(s_1^*; r_2^*)$.
  - Set $x^* = (pk, d^*, c^*, i)$, $\underline{w^* = (0^k, s, sk)}$. Then compute $e^* = \mathsf{NIWI}.\mathcal{P}(1^k, x^*, w^*, \mathcal{L}_{\mathsf{NIWI}}; r_3^*)$.
  - Generate program $\widetilde{P}^{i-1}_{pk,K,\mathcal{D},\mathcal{L},(x^*,e^*)}$ defined in Figure 3.
  - Compute $\widetilde{P} = \mathsf{iO}.\mathsf{Obf}(\widetilde{P}^{i-1}_{pk,K,\mathcal{D},\mathcal{L},(x^*,e^*)}; R)$.
  - Output $(pk, \widetilde{P})$.

Claim 4.4. *For every (non-uniform) $\mathsf{poly}(2^{\kappa^c})$-sized distinguisher $\mathcal{A}$ there exists a negligible function $\mu(\cdot)$ such that:*

$$\left| \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,4}) = 1] - \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,5}) = 1] \right| = \mu(2^{k^\epsilon})$$

Proof. We will prove this claim based on the sub-exponential witness indistinguishability of the NIWI against non-uniform adversaries. Towards a contradiction, suppose there exists a distinguisher $\mathcal{A}$ and a polynomial $p(\cdot)$ such that

$$\left| \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,4}) = 1] - \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,5}) = 1] \right| \geq \frac{1}{p(2^{k^\epsilon})}$$

Then there exists a non-uniform adversary $\mathcal{A}'$ that non-uniformly fixes $s_1^* = f^{-1}(i)$, then begins the experiment. It obtains e^* from the CPA challenger either using witness $w^* = (r_1^*, 0^{2k})$ as in $\mathsf{Hybrid}_{i-1,4}$, or using witness $w^* = (0^k, s, sk)$ as in $\mathsf{Hybrid}_{i-1,5}$. It completes the rest of the experiment according to $\mathsf{Hybrid}_{i-1,4}$ except setting e^* according to the NIWI obtained from the external challenger. It implements the oracle $\mathbb{O}$ for $\mathcal{A}$ (this requires size 2^{κ^c}). It then mirrors the output of $\mathcal{A}$ given the resulting distribution, which implies that

$$\left| \Pr[\mathcal{A}'(w^* = (r_1^*, 0^{2k})) = 1] - \Pr[\mathcal{A}'(w^* = (0^k, s, sk)) = 1] \right| \geq \frac{1}{p(2^{k^\epsilon})}$$

which gives a contradiction, as desired. Therefore, by sub-exponential security of the NIWI (with security parameter k), for every $i \in [1, 2^\kappa]$ and every (non-uniform) $\mathsf{poly}(2^{\kappa^c})$-sized distinguisher $\mathcal{A}$ there exists a negligible function $\mu_4(\cdot)$ such that:

$$\left| \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,4}) = 1] - \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,5}) = 1] \right| \leq \mu_4(2^{k^\epsilon})$$

□

- $\mathsf{Hybrid}_{i-1,6}$ is defined as follows: (we underline the difference between $\mathsf{Hybrid}_{i-1,5}$ and $\mathsf{Hybrid}_{i-1,6}$)
  - Set $k = \kappa^{\frac{1}{\epsilon}}$.
  - Sample $s \leftarrow \{0,1\}^k$ and set $(pk, sk) \leftarrow \mathsf{KeyGen}(s)$.
  - Sample $K \leftarrow \{0,1\}^k, R \leftarrow \{0,1\}^k$.
  - Set $(r_1^*, r_2^*, r_3^*) \leftarrow \{0,1\}^{2\kappa+k}$.
  - Compute (in time upto 2^κ) value s_1^* such that $f(s_1^*) = i$.
  - Set $\underline{d^* = \mathcal{D}_1(r_1^*)}$ and $c^* = \mathsf{Enc}_{pk}(s_1^*; r_2^*)$.
  - Set $x^* = (pk, d^*, c^*, i)$, $w^* = (0^\kappa, s, sk)$. Then compute $e^* = \mathsf{NIWI}.\mathcal{P}(1^k, x^*, w^*, \mathcal{L}_{\mathsf{NIWI}}; r_3^*)$.
  - Generate program $\widetilde{P}^{i-1}_{pk,K,\mathcal{D},\mathcal{L},(x^*,e^*)}$ defined in Figure 3.
  - Compute $\widetilde{P} = \mathsf{iO}.\mathsf{Obf}(\widetilde{P}^{i-1}_{pk,K,\mathcal{D},\mathcal{L},(x^*,e^*)}; R)$.
  - Output $(pk, \widetilde{P})$.

Claim 4.5. *For every (non-uniform) $\mathsf{poly}(2^{\kappa^c})$-sized distinguisher $\mathcal{A}$ there exists a negligible function $\mu(\cdot)$ such that:*

$$\left| \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,5}) = 1] - \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,6}) = 1] \right| = \mu(2^{k^\epsilon})$$

Proof. We will prove this claim based on 2^{k^δ}-hardness of $(\mathcal{D}_0, \mathcal{D}_1)$ against non-uniform adversaries. Towards a contradiction, suppose there exists a distinguisher $\mathcal{A}$ and a polynomial $p(\cdot)$ such that

$$\left| \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,5}) = 1] - \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,6}) = 1] \right| \geq \frac{1}{p(2^{k^\epsilon})}$$

Then there exists a (non-uniform) adversary $\mathcal{A}'$ that non-uniformly fixes $s_1^* = f^{-1}(i)$, then begins the experiment. It obtains d^* from an external challenger sampled either as $\mathcal{D}_0(r^*)$ as in $\mathsf{Hybrid}_{i-1,5}$, or as $\mathcal{D}_1(r^*)$ as in $\mathsf{Hybrid}_{i-1,6}$. It completes the rest of the experiment according to $\mathsf{Hybrid}_{i-1,5}$ except setting d^* according to the sample obtained from the external challenger. It then mirrors the output of $\mathcal{A}$ given the resulting distribution, which implies that

$$\left| \Pr[\mathcal{A}'(d^* = \mathcal{D}_0(r^*)) = 1] - \Pr[\mathcal{A}'(d^* = \mathcal{D}_1(r^*)) = 1] \right| \geq \frac{1}{p(2^{k^\epsilon})} \geq \frac{1}{p(2^{k^\delta})}$$

which follows because $\epsilon < \delta$, and gives a contradiction to the 2^{k^δ}-hardness of $(\mathcal{D}_0, \mathcal{D}_1)$, as desired.

Therefore, by sub-exponential indistinguishability between the distributions $(\mathcal{D}_0, \mathcal{D}_1)$ (with security parameter k), for every $i \in [1, 2^\kappa]$ and every (non-uniform) $\mathsf{poly}(2^{\kappa^c})$-sized distinguisher $\mathcal{A}$ there exists a negligible function $\mu_5(\cdot)$ such that:

$$\left| \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,5}) = 1] - \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,6}) = 1] \right| \leq \mu_5(2^{k^\epsilon})$$

□

- $\mathsf{Hybrid}_{i-1,7}$ is defined as follows: (we underline the difference between $\mathsf{Hybrid}_{i-1,6}$ and $\mathsf{Hybrid}_{i-1,7}$)
  - Set $k = \kappa^{\frac{1}{\epsilon}}$.
  - Sample $s \leftarrow \{0,1\}^k$ and set $(pk, sk) \leftarrow \mathsf{KeyGen}(s)$.
  - Sample $K \leftarrow \{0,1\}^k, R \leftarrow \{0,1\}^k$.
  - Set $(r_1^*, r_2^*, r_3^*) \leftarrow \{0,1\}^{2\kappa+k}$.
  - Compute (in time upto 2^κ) value s_1^* such that $f(s_1^*) = i$.
  - Set $d^* = \mathcal{D}_1(r_1^*)$ and $c^* = \mathsf{Enc}_{pk}(s_1^*; r_2^*)$.
  - Set $x^* = (pk, d^*, c^*, i)$, $\underline{w^* = (r_1^*, 0^{2k})}$. Then compute $e^* = \mathsf{NIWI}.\mathcal{P}(1^k, x^*, w^*, \mathcal{L}_{\mathsf{NIWI}}; r_3^*)$.
  - Generate program $\widetilde{P}^{i-1}_{pk,K,\mathcal{D},\mathcal{L},(x^*,e^*)}$ defined in Figure 3.
  - Compute $\widetilde{P} = \mathsf{iO}.\mathsf{Obf}(\widetilde{P}^{i-1}_{pk,K,\mathcal{D},\mathcal{L},(x^*,e^*)}; R)$.
  - Output $(pk, \widetilde{P})$.

Claim 4.6. *For every (non-uniform) $\mathsf{poly}(2^{\kappa^c})$-sized distinguisher $\mathcal{A}$ there exists a negligible function $\mu(\cdot)$ such that:*

$$\left|\Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,6}) = 1] - \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,7}) = 1]\right| = \mu(2^{k^\epsilon})$$

Proof. By sub-exponential witness indistinguishability of the NIWI against non-uniform adversaries (and following an identical argument to that of the indistinguishability between $\mathsf{Hybrid}_{i-1,4}$ and $\mathsf{Hybrid}_{i-1,5}$), for every $i \in [1, 2^\kappa]$ and every (non-uniform) $\mathsf{poly}(2^{k^\epsilon})$-sized distinguisher $\mathcal{A}$ there exists a negligible function $\mu_6(\cdot)$ such that:

$$\left|\Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,6}) = 1] - \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,7}) = 1]\right| \leq \mu_6(2^{k^\epsilon})$$

□

- $\mathsf{Hybrid}_{i-1,8}$ is defined as follows: (we underline the difference between $\mathsf{Hybrid}_{i-1,7}$ and $\mathsf{Hybrid}_{i-1,8}$)
  - Set $k = \kappa^{\frac{1}{\epsilon}}$.
  - Sample $s \leftarrow \{0,1\}^k$ and set $(pk, sk) \leftarrow \mathsf{KeyGen}(s)$.
  - Sample $K \leftarrow \{0,1\}^k, R \leftarrow \{0,1\}^k$.
  - Set $(r_1^*, r_2^*, r_3^*) \leftarrow \{0,1\}^{2\kappa+k}$.
  - Do not compute value s_1^* such that $f(s_1^*) = i$.
  - Set $d^* = \mathcal{D}_1(r_1^*)$ and $\underline{c^* = \mathsf{Enc}_{pk}(0^\kappa; r_2^*)}$.
  - Set $x^* = (pk, d^*, c^*, i)$, $w^* = (r_1^*, 0^{2k})$. Then compute $e^* = \mathsf{NIWI}.\mathcal{P}(1^k, x^*, w^*, \mathcal{L}_{\mathsf{NIWI}}; r_3^*)$.
  - Generate program $\widetilde{P}^{i-1}_{pk,K,\mathcal{D},\mathcal{L},(x^*,e^*)}$ defined in Figure 3.
  - Compute $\widetilde{P} = \mathsf{iO}.\mathsf{Obf}(\widetilde{P}^{i-1}_{pk,K,\mathcal{D},\mathcal{L},(x^*,e^*)}; R)$.
  - Output $(pk, \widetilde{P})$.

Claim 4.7. *For every (non-uniform) $\mathsf{poly}(2^{\kappa^c})$-sized distinguisher $\mathcal{A}$ there exists a negligible function $\mu(\cdot)$ such that:*

$$\left|\Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,7}) = 1] - \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,8}) = 1]\right| = \mu(2^{k^\epsilon})$$

Proof. By sub-exponential IND-CPA security of the encryption scheme against non-uniform adversaries (and following an identical argument to that of the indistinguishability between $\mathsf{Hybrid}_{i-1,3}$ and $\mathsf{Hybrid}_{i-1,4}$), for every $i \in [1, 2^\kappa]$ and every (non-uniform) $\mathsf{poly}(2^{\kappa^c})$-sized distinguisher $\mathcal{A}$ there exists a negligible function $\mu_7(\cdot)$ such that:

$$\left| \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,3}) = 1] - \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,4}) = 1] \right| \leq \mu_7(2^{k^\epsilon})$$

□

- $\mathsf{Hybrid}_{i-1,9}$ is identical to $\mathsf{Hybrid}_{i-1,8}$ except that $(r_1^*, r_2^*, r_3^*) = \mathsf{PRF}(K, i)$.

  Claim 4.8. *For every (non-uniform) $\mathsf{poly}(2^{\kappa^c})$-sized distinguisher $\mathcal{A}$ there exists a negligible function $\mu(\cdot)$ such that:*

  $$\left| \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,8}) = 1] - \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,9}) = 1] \right| = \mu(2^{k^\epsilon})$$

  Proof. By sub-exponential security of the puncturable PRF (with security parameter k), we have directly that for every $i \in [1, 2^\kappa]$ and every (non-uniform) $\mathsf{poly}(2^{\kappa^c})$-sized distinguisher $\mathcal{A}$ there exists a negligible function $\mu_8(\cdot)$ such that:

  $$\left| \Pr[\mathcal{A}(\mathsf{Hybrid}_{i-1,8}) = 1] - \Pr[\mathcal{A}(\mathsf{Hybrid}_{i-1,9}) = 1] \right| \leq \mu_8(2^{k^\epsilon})$$

  □

- $\mathsf{Hybrid}_{i-1,10} \equiv \mathsf{Hybrid}_i$.

  Claim 4.9. *For every (non-uniform) $\mathsf{poly}(2^{\kappa^c})$-sized distinguisher $\mathcal{A}$ there exists a negligible function $\mu(\cdot)$ such that:*

  $$\left| \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,9}) = 1] - \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,10}) = 1] \right| = \mu(2^{k^\epsilon})$$

  Proof. Note that now the programs $P^i_{pk,K,\mathcal{D},\mathcal{L}}$ and $\widetilde{P}^{i-1}_{pk,K,\mathcal{D},\mathcal{L},(x^*,e^*)}$ have identical functionality. Therefore, by sub-exponential security of indistinguishability obfuscation (with security parameter k), for every $i \in [1, 2^\kappa]$ and every (non-uniform) $\mathsf{poly}(2^{\kappa^c})$-sized distinguisher $\mathcal{A}$ there exists a negligible function $\mu_9(\cdot)$ such that:

  $$\left| \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,9}) = 1] - \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1,10}) = 1] \right| \leq \mu_9(2^{k^\epsilon})$$

  □

By combining all the above claims, we have that for every $i \in [1, 2^\kappa]$ and every (non-uniform) $\mathsf{poly}(2^{\kappa^c})$-sized distinguisher $\mathcal{A}$ there exists a negligible function $\mu(\cdot)$ such that:

$$\left| \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{i-1}) = 1] - \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_i) = 1] \right| \leq \mu(2^{k^\epsilon})$$

Since $k^\epsilon = \kappa^c$, we therefore have that for every (non-uniform) $\mathsf{poly}(2^{\kappa^c})$-sized distinguisher $\mathcal{A}$ (and in particular also for every $\mathsf{poly}(\kappa)$-sized distinguisher $\mathcal{A}$) there exists a negligible function μ' such that:

$$\left| \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_0) = 1] - \Pr[\mathcal{A}^{\mathbb{O}}(\mathsf{Hybrid}_{2^\kappa}) = 1] \right| \leq 2^\kappa \mu(2^{\kappa^c}) = \mu'(\kappa)$$

□

5 Constructing COA Secure Obfuscation

We present and analyze our construction of COA obfuscators. Our construction actually shows how to fortify functionality-preserving obfuscators with any underlying hiding property. COA obfuscators as in Definition 3.3 are obtained as a corollary.

Construction 5.1. *We require the following primitives:*

- *Let* ccacom *denote an $\epsilon(\kappa)$-secure CCA commitment scheme according to Definition 2.3 and let $\mathbb{O}$ denote the (deterministic, inefficient) oracle that implements the* CCA.DeCom *algorithm for* ccacom*. That is, on input a commitment string* com*, the oracle $\mathbb{O}$ outputs either a message $m \in \{0,1\}^*$ or $\perp$. Also, let $T = \mathsf{poly}(|m|, 2^\kappa)$ (where $|m|$ denotes the size of message space for* ccacom*).*
- *Let* r-NIDI *denote a robust NIDI w.r.t. oracle $\mathbb{O}$ for language $\mathcal{L}_\phi$, defined below.*
- *Let $\mathcal{O}$ denote the underlying obfuscator for our COA fortification. We will assume that this obfuscator is secure against $\mathsf{poly}(T)$-sized adversaries. This can be achieved by appropriately scaling the security parameter for $\mathcal{O}$, since $\mathcal{O}$ is assumed to be subexponentially secure.*
- *Define language $\mathcal{L}_\phi = \{\{O, c\} : \exists (C, r_1, r_2) : O = \mathcal{O}(C; r_1) \wedge c = \mathsf{ccacom}(C; r_2) \wedge \phi(C) = 1\}$*

The algorithm $c\mathcal{O}.\mathsf{Obf}(1^\kappa, C, \phi)$ does the following:

- *Define distribution $\mathcal{D}_C(r_1||r_2) = \{\mathcal{O}(C; r_1), c = \mathsf{ccacom}(C; r_2)\}$ for uniformly sampled r_1, r_2.*
- *Output $\pi \leftarrow$ r-NIDI.$\mathcal{P}(1^\kappa, \mathcal{D}_C, \mathcal{L}_\phi)$ computed using uniform randomness r_c.*

The algorithm $c\mathcal{O}.\mathsf{Ver}(1^\kappa, \widehat{C}, \phi)$ does the following:

- *Sample randomness $r_{\mathcal{R}}$.*
- *Output $y \leftarrow$ r-NIDI.$\mathcal{V}(1^\kappa, \pi; r_R)$.*

We show:

Theorem 5.1. *For any $(T(\kappa), \epsilon(\kappa))$, if the underlying commitment scheme is $\epsilon(\kappa)$-CCA-secure as in Definition 2.3 and with decommitment oracle that's implementable in time $T(\kappa)$, and the underlying NIDI is $\epsilon(\kappa)$-gap robust w.r.t. the decommitment oracle for the CCA commitments (see Definition 4.1), then the above construction is a $(T(\kappa), \epsilon(\kappa)/4)$-gap COA fortification for any injective obfuscation as in Definition 3.5.*

Corollary 5.1. *Assuming the existence of sub-exponentially secure one-way functions and sub-exponentially secure indistinguishability obfuscation, there exists COA-secure obfuscation for all polynomial-sized circuits, satisfying Definition 3.3.*

Proof. (Sketch) By [7], assuming the existence of sub-exponentially secure iO and sub-exponentially secure puncturable PRFs, there exist subexponentially secure piO schemes for any polynomial sized circuit family. That is, there exists a constant $\delta > 0$ such that for $T = 2^{\kappa^\delta}$, and every $\mathsf{poly}(T)$-sized distinguisher $\mathcal{D}$, $\mathsf{Adv}^{\mathsf{Samp}}_{\mathsf{piO},\mathcal{D}} = \mathsf{negl}(T)$ where Samp is an admissible sampler according to Definition 2.4.[6]

Furthermore, for $\epsilon(\kappa) = 2^{-o(\log^c(\kappa))}$ and some constant $c > 1$, there exist $\epsilon(\kappa)$-secure CCA commitments satisfying Definition 2.3 for which the decommitment oracle can be implemented in time $T(\kappa)$, and by Theorem 4.1 there exist robust NIDI arguments satisfying $\epsilon(\kappa)$ gap distributional indistinguishability w.r.t. the decommitment oracle for the CCA commitments. Then, the theorem above implies that there exists $(2^{\kappa^\delta}, 2^{-o(\log^c(\kappa))})$-gap COA fortification for any injective obfuscation and in particular, for the injective piO

[6] This scheme can be made injective by attaching a perfectly binding commitment of the circuit to its original obfuscation, but notice that this step is in fact not necessary: the CCA commitment which is part of the fortification process provides sufficient injectivity for the COA game to make sense and for the proof to go through.

scheme described above. This results in a COA-secure obfuscation scheme, whose correctness and verifiability are immediate from those of the COA fortification. Furthermore, by definition of fortification, this means there is a T-sized transformation $\mathcal{T}$ on distinguishers such that for any admissible sampler Samp and distinguisher $\mathcal{D}$,

$$\mathsf{Adv}^{\mathsf{Samp}}_{\mathcal{O},\mathcal{T}(\mathcal{D})} \geq \epsilon(\kappa) \cdot \mathsf{COAAdv}^{\mathsf{Samp}}_{c\mathcal{O},\mathcal{D}}$$

This implies that for any $\mathcal{T}$-sized distinguisher $\mathcal{D}$, $\mathsf{COAAdv}^{\mathsf{Samp}}_{c\mathcal{O},\mathcal{D}} = \mathsf{negl}(\kappa)$. □

Now, assume that for $\epsilon(\kappa) = 2^{-o(\log^c(\kappa))}$ and some constant $c > 1$, there exist $\epsilon(\kappa)$-secure CCA commitments satisfying Definition 2.3 for which the decommitment oracle can be implemented in time $T(\kappa)$ where $T(\kappa) = 2^{\kappa^\delta}$ for some constant $\delta > 0$, and by Theorem 4.1 there exist robust NIDI arguments satisfying $\epsilon(\kappa)$ gap distributional indistinguishability w.r.t. the decommitment oracle for the CCA commitments. Then, the theorem above implies that there exist $(2^{\kappa^\delta}, 2^{-o(\log^c(\kappa))})$-gap COA fortification for any injective obfuscation. Furthermore, the corollary implies that if in addition there exist subexponential IO and one way functions then there exist COA-secure obfuscators for circuits.

5.1 Proof of Theorem 5.1

The proof proceeds via the following sequence of hybrid games.

Hybrid_0 : This hybrid corresponds to the COA obfuscation security game with $b = 1$. For any ϕ-satisfying sampler Samp let $\mathsf{Hyb}_0\mathsf{Adv}^{\mathsf{Samp}}_{c\mathcal{O},\mathcal{D}}$ denote the probability that the oracle-aided adversary outputs 1 in this experiment, which is the term

$$q^{\mathsf{Samp},1}_{c\mathcal{O},\mathcal{D}} := \Pr_{\substack{(C_1,C_2,z)\leftarrow\mathsf{Samp}(1^\kappa)\\ \widehat{C}\leftarrow c\mathcal{O}.\mathsf{Obf}(1^\kappa,C_1)}}\left[\mathcal{D}^{\mathcal{O}^{-1}\circ c\mathcal{O}.\mathsf{Ver}\circ \mathsf{Filt}_{\widehat{C}}}(\widehat{C},z)=1\right]$$

from Definition 3.5.

Hybrid_1 : This hybrid is identical to Hybrid_0, except that the deobfuscation oracle executes $\mathsf{r\text{-}NIDI}.\mathcal{V}(1^k,\pi_k)$ and obtains output either $\perp$, or a pair $(\widetilde{C},c)$ where $\widetilde{C}$ is an obfuscated circuit and c is a non-malleable commitment string. If $\mathsf{r\text{-}NIDI}.\mathcal{V}(1^k,\pi_k)$ outputs $\perp$, the deobfuscation oracle returns $\perp$ to the adversary. Otherwise, it invokes the decommitment oracle $\mathrm{CCA}.\mathsf{DeCom}$ on c. Let $\mathsf{Hyb}_1\mathsf{Adv}^{\mathsf{Samp}}_{c\mathcal{O},\mathcal{D}}$ denote the probability that the oracle-aided adversary outputs 1 in this experiment.

Claim 5.1. *For every (non-uniform)* $\mathsf{poly}(\kappa)$*-sized adversary* $\mathcal{A}$*, there exists a negligible function* $\mu(\cdot)$ *such that for large enough* $\kappa \in \mathbb{N}$*,*

$$|\mathsf{Hyb}_1\mathsf{Adv}^{\mathsf{Samp}}_{c\mathcal{O},\mathcal{D}} - \mathsf{Hyb}_0\mathsf{Adv}^{\mathsf{Samp}}_{c\mathcal{O},\mathcal{D}}| \leq \mu(\kappa)$$

Proof. Suppose the claim is not true. We will prove that this hybrid is indistinguishable from previous hybrid due to soundness of the robust NIDI, as per Definition 4.1, and perfect injectivity of the obfuscation and non-malleable commitment. Indeed, suppose there exists a polynomial $p(\cdot)$ such that

$$|\mathsf{Hyb}_1\mathsf{Adv}^{\mathsf{Samp}}_{c\mathcal{O},\mathcal{D}} - \mathsf{Hyb}_0\mathsf{Adv}^{\mathsf{Samp}}_{c\mathcal{O},\mathcal{D}}| \geq \frac{1}{p(\kappa)}$$

Because the two games are identical except one inverts the obfuscation and the other inverts the commitment, denoting $\mathcal{D}$'s oracle queries by $\widehat{C}_1, \widehat{C}_2, \ldots, \widehat{C}_n$,

$$\Pr[\exists i \in [n] \text{ s.t. } c\mathcal{O}.\mathsf{Ver}(\widehat{C}_i) \neq \perp \bigwedge \mathsf{r\text{-}NIDI}.\mathcal{V}(\widehat{C}_i) = (\widetilde{C},c)$$
$$\bigwedge \nexists(C,r_1,r_2) \text{ s.t. } \widetilde{C} = \mathcal{O}(C;r_1), c = \mathsf{ccacom}(C;r_2)] \geq \frac{1}{p(\kappa)}$$

This contradicts soundness of the $\mathsf{r\text{-}NIDI}$. □

Hybrid_2 : This hybrid is identical to Hybrid_1, except that the challenger sets $b = 2$.

The difference between this game and Hybrid_1 is that in Hybrid_d for $d \in \{1, 2\}$, the adversary obtains $\mathsf{r\text{-}NIDI}.\mathcal{P}(1^\kappa, \mathcal{D}_{C_d})$.

Claim 5.2. *For every (non-uniform)* $\mathsf{poly}(\kappa)$*-sized adversary* $\mathcal{A}$*, there exists a negligible function* $\mu(\cdot)$ *such that for large enough* $\kappa \in \mathbb{N}$*,*

$$|\mathsf{Hyb}_2\mathsf{Adv}^{\mathsf{Samp}}_{c\mathcal{O},\mathcal{D}} - \mathsf{Hyb}_1\mathsf{Adv}^{\mathsf{Samp}}_{c\mathcal{O},\mathcal{D}}| \leq \mu(\kappa)$$

Proof. Suppose towards a contradiction that there exists a $\mathsf{poly}(\kappa)$-sized adversary $\mathcal{A}$ and polynomial $p(\cdot)$ such that for large enough $\kappa \in \mathbb{N}$,

$$|\mathsf{Hyb}_2\mathsf{Adv}^{\mathsf{Samp}}_{c\mathcal{O},\mathcal{D}} - \mathsf{Hyb}_1\mathsf{Adv}^{\mathsf{Samp}}_{c\mathcal{O},\mathcal{D}}| \geq \frac{1}{p(\kappa)}$$

By the ϵ-gap robust indistinguishability of r-NIDI w.r.t. the decommitment oracle $\mathbb{O}$, we have that for every $\mathsf{poly}(\kappa)$-size distinguisher D with

$$\left| \Pr[D^{\mathbb{O}}(\mathcal{P}(1^\kappa, \mathcal{D}_{C_1})) = 1] - \Pr[D^{\mathbb{O}}(\mathcal{P}(1^\kappa, \mathcal{D}_{C_2})) = 1] \right| \geq \frac{1}{p(\kappa)}$$

there exists a distinguisher $\widehat{D} = \mathcal{T}(D)$ of size $\mathsf{poly}(\kappa)$ that satisfies:

$$\left| \Pr[\widehat{D}^{\mathbb{O}}(\mathcal{X}_1) = 1] - \Pr[\widehat{D}^{\mathbb{O}}(\mathcal{X}_2) = 1] \right| \geq \frac{\epsilon(\kappa)}{p(\kappa)},$$

where $\mathcal{X}_1 = \mathcal{O}(C_1; r), \mathsf{ccacom}(C_1; r')$ and $\mathcal{X}_2 = \mathcal{O}(C_2; r), \mathsf{ccacom}(C_2; r')$ where (r, r') are sampled uniformly.

We will now prove that $\widehat{\mathcal{D}}^{\mathbb{O}}$ either contradicts security of the CCA commitment or that (T, ϵ)-Gap security of the COA holds. To see this, consider an intermediate distribution $\mathcal{X}_{12} = \mathcal{O}(C_1; r), \mathsf{ccacom}(C_2; r')$ where (r, r') are sampled uniformly. By the above statement, we have that either

$$\left| \Pr[\widehat{D}^{\mathbb{O}}(\mathcal{X}_1) = 1] - \Pr[\widehat{D}^{\mathbb{O}}(\mathcal{X}_{12}) = 1] \right| \geq \frac{\epsilon(\kappa)}{2p(\kappa)},$$

or

$$\left| \Pr[\widehat{D}^{\mathbb{O}}(\mathcal{X}_{12}) = 1] - \Pr[\widehat{D}^{\mathbb{O}}(\mathcal{X}_2) = 1] \right| \geq \frac{\epsilon(\kappa)}{2p(\kappa)},$$

In the former case, $\widehat{D}$ contradicts the CCA security of ccacom. In the latter case, since the oracle $\mathbb{O}$ can be implemented in time T, $\widehat{D}$ distinguishes the underlying obfuscations. □

Hybrid_3 : This hybrid corresponds to the COA obfuscation security game with $b = 2$.

Note that Hybrid_3 is identical to Hybrid_2, except that the deobfuscation oracle executes $\mathsf{r\text{-}NIDI}.\mathcal{V}(1^k, \pi_k)$ and obtains output either $\perp$, or a pair $(\widetilde{C}, c)$ where $\widetilde{C}$ is an obfuscated circuit and c is a non-malleable commitment string. If $\mathsf{r\text{-}NIDI}.\mathcal{V}(1^k, \pi_k)$ outputs $\perp$, the deobfuscation oracle returns $\perp$ to the adversary. Otherwise, it invokes the deobfuscation oracle $\mathcal{O}^{-1}$ on $\widetilde{C}$.

Claim 5.3. *For every (non-uniform)* $\mathsf{poly}(\kappa)$*-sized adversary* $\mathcal{A}$*, there exists a negligible function* $\mu(\cdot)$ *such that for large enough* $\kappa \in \mathbb{N}$*,*

$$|\mathsf{Hyb}_3\mathsf{Adv}^{\mathsf{Samp}}_{c\mathcal{O},\mathcal{D}} - \mathsf{Hyb}_2\mathsf{Adv}^{\mathsf{Samp}}_{c\mathcal{O},\mathcal{D}}| \leq \mu(\kappa)$$

Proof. Because the two games are identical except one inverts the obfuscation and the other inverts the commitment, the proof of this claim follows by the soundness of r-NIDI, identically to the indistinguishability claim between Hybrid_0 and Hybrid_1. □

We conclude that the transform from the proof of indistinguishability between $\mathsf{Hyb}_2\mathsf{Adv}^{\mathsf{Samp}}_{c\mathcal{O},\mathcal{D}}$ and $\mathsf{Hyb}_1\mathsf{Adv}^{\mathsf{Samp}}_{c\mathcal{O},\mathcal{D}}$ results in an adversary $\widehat{D}^{\mathbb{O}}$ that runs in time T, and achieves at least an advantage of $\epsilon/4$ in distinguishing obfuscated programs. This implies $(T, \epsilon/4)$-gap security of the fortification.

5.2 Proof of verifiability

Since $c\mathcal{O}.\mathsf{Ver}(\pi, \mathcal{L}_\phi) = \text{r-NIDI}.\mathcal{V}(1^\kappa, \pi; r_R)$ for uniform $r_\mathcal{R}$, and since, by soundness of r-NIDI, for every ensemble of polynomial-length strings $\{\Pi_\kappa\}_\kappa$ there exists a negligible function μ such that

$$\Pr_{x \leftarrow \text{r-NIDI}.\mathcal{V}(1^\kappa, \Pi_\kappa, \mathcal{L}_\phi)} \Big[(x \neq \bot) \wedge (x \notin \mathcal{L}_\phi)] \leq \mu(\kappa),$$

the verifiability of $c\mathcal{O}$ follows immediately.

6 Completely CCA-secure Encryption

In the context of public-key encryption schemes, non-malleability is traditionally aimed to prevent situations whereby an attacker, given "honestly generated" public encryption key pk and ciphertext $c = Enc(pk, m, r)$ for an some message m and randomness r, manages to generate a ciphertext c' such that $Dec(sk, c') = m'$, where sk is the secret key associated with pk and m' is related to m in some predefined way. Security against chosen ciphertext attacks (CCA2) extends non-malleability by preventing situations whereby an attacker creates *any* new ciphertext c' such that learning $m' = Dec(sk, c')$ would enable the attacker to break semantic security of the scheme - regardless of how (or whether) m' relates to some previously encrypted m.

Fischlin [11] (and later also Ventre and Visconti [29]) considered a natural extension of non-malleable public-key encryption, which prevents situations whereby an attacker , given pk and ciphertext $c = Enc(pk, m, r)$, manages to generate a *pair* (pk', c') that "look legitimate", and furthermore "c' is an encryption of m' under public key pk', whereas m' relates to m in some predefined way. Note that the clause "c' is an encryption of m'" may be interpreted in a number of ways, and may not even be always well defined. Indeed, Fischlin and Ventre and Visconti further provide a number of possible interpretations and study the resulting notions. Furthermore, their primary notion, where "c' is an encryption of m'" means "there exists r such that $c' = Enc(pk', m', r)$", is shown to imply that such encryption schemes can double up as non-malleable commitment schemes. On the down side, without additional set-up assumptions, this notion is shown to be uninstantiable by way of any reduction to a standard (polynomial-time) game-based assumption.

C-CCA encryption aims to extend completely non-malleable encryption in the same way that CCA2 encryption extends non-malleable encryption. (Alternatively, C-CCA encryption extends CCA2 encryption in the same way that complete non-malleable encryption extends non-malleable encryption.) This is done as follows:

First, to avoid edge cases where "c' is an encryption of m'" is undefined (e.g. encryption schemes that allow any message to result in any ciphertext with a tiny-but-positive probability), we require the encryption algorithm to explicitly decide whether a purported public-key is legitimate; furthermore, ciphertexts that were generated honestly using public key that was recognized as legitimate should be decrypted correctly - namely any public key pk that's recognized as legitimate should be associated with a legitimate secret key sk such that $Dec(sk, Enc(pk, m, r)) = m$. Note that that this should hold even if pk is not in the image of the key generation algorithm.

Next, we extend CCA2 security to prevent even those situations whereby an attacker creates a new *pair* (pk', c') such that pk' is considered legitimate and learning $m' = Dec(sk, c')$ would enable the attacker to break semantic security of the scheme. As in the case of CCA2, this holds regardless of how (or whether) m' relates to some previously encrypted m.

While we do not know whether C-CCA formally implies Fischlin's complete non-malleability (mainly due to those definitional edge cases), it is significantly more powerful. In particular, it implies a variant of complete CCA commitment, and is susceptible to the same uninstantiability results in the plain model. (Our construction bypasses these uninstantiability due to the use of subexponentially hard primitives.)

6.1 Defining completely CCA-secure encryption

Definition 6.1 (C-CCA-security)**.** An encryption scheme $\mathcal{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is Completely CCA (C-CCA) secure for a message-space family $\mathcal{M} = \{\mathcal{M}_\kappa\}_\kappa$, if there is a negligible function $\mu(\cdot)$ such that:

Correctness: $\Pr_{(\mathsf{pk},\mathsf{sk})\leftarrow\mathsf{KeyGen}(1^\kappa)}[\mathsf{Dec}(\mathsf{sk},\mathsf{Enc}(\kappa,\mathsf{pk},m)) = m] > 1-\mu(\kappa)$ for all $\kappa \in \mathbf{N}$, $m \in \mathcal{M}_\kappa$. Furthermore, we allow Enc to output a special failure symbol $\perp$, such that $\mathsf{Dec}(\mathsf{sk},\perp) = \perp$ for all sk, and $\perp \notin \mathcal{M}$.

Unique decryptability: A string $\mathsf{pk} \in \{0,1\}^{poly(\kappa)}$ is *useless* if $\Pr[\mathsf{Enc}(\kappa,\mathsf{pk},m) \neq \perp] \leq \mu(\kappa)$ for any $m \in \mathcal{M}_\kappa$. Then, for any $\mathsf{pk} \in \{0,1\}^{poly(\kappa)}$ that is not useless there is a *unique* sk such that $\Pr[\mathsf{Dec}(\mathsf{sk},\mathsf{Enc}(\kappa,\mathsf{pk},m)) = m] \geq 1-\mu(\kappa)$ for all $m \in \mathcal{M}_\kappa$, and furthermore $(\mathsf{pk}',\mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa, r)$ for some r, pk'. We call sk the *opening* of pk.

Complete CCA security: A function $\mathcal{D}_{\kappa,\mathsf{pk}^*,c^*}(\mathsf{pk},c)$ is a *valid C-CCA decryption oracle* for $\mathcal{PKE}$ if whenever $(\mathsf{pk},c) \neq (\mathsf{pk}^*,c^*)$ and pk is not useless, $\mathcal{D}_{\kappa,\mathsf{pk}^*,c^*}(\mathsf{pk},c) = \mathsf{Dec}(\mathsf{sk},c)$, where sk is the opening of pk.

Then there exists a valid C-CCA decryption oracle $\mathcal{D}$ such that for any PPT adversary $\mathcal{A}$, for all sufficiently large κ,

$$\mathsf{Adv}^{\text{c-cca}}_{\mathcal{PKE},\mathcal{A},\mathcal{D}}(\kappa) := \left|\Pr[\mathsf{Exp}^{\text{c-cca}}_{\mathcal{PKE},\mathcal{A},\mathcal{D}}(\kappa) = 1] - \frac{1}{2}\right| < \mu(\kappa),$$

where $\mathsf{Exp}^{\text{c-cca}}_{\mathcal{PKE},\mathcal{A},\mathcal{D}}(\kappa)$ denotes the following experiment:

1. Let $(\mathsf{pk}^*,\mathsf{sk}^*) \leftarrow \mathsf{KeyGen}(1^\kappa)$.
2. Let $(m_0,m_1,s) \leftarrow \mathcal{A}^{\mathcal{D}_{\kappa,\mathsf{pk}^*,\perp}}(\mathsf{pk}^*)$.
3. Let $c^* \leftarrow \mathsf{Enc}(\kappa,\mathsf{pk}^*,m_{b^*})$ where $b^* \leftarrow_\$ \{0,1\}$.
4. Let $b' \leftarrow \mathcal{A}^{\mathcal{D}_{\kappa,\mathsf{pk}^*,c^*}}(s,c^*)$.
5. Return 1 if $b' = b^*$.

6.2 C-CCA secure PKE in the Plain model

The starting point of our C-CCA secure PKE scheme is the iO-based CCA-secure encryption scheme of Sahai and Waters [28]. We first instantiate that scheme with a COA obfuscator (with an appropriate correctness predicate). Next, we add to the encryption algorithm a simple sanity check that allows recognizing and rejecting "obviously bogus" public keys.

Construction 6.1. *Let:*

- $\mathcal{M}_\kappa = \{0,1\}^{\ell(\kappa)}$.
- $F_1 : \{0,1\}^{\eta(\kappa)} \times \{0,1\}^{2\kappa} \to \{0,1\}^{\ell(\kappa)}$ *and* $F_2 : \{0,1\}^{\eta(\kappa)} \times \{0,1\}^{2\kappa+\ell(\kappa)} \to \{0,1\}^{\tau(\kappa)}$ *be two puncturable pseudo-random function families, with* $\eta(\kappa)$*-bit keys for security parameter* κ*, that have* 2^{κ^ϵ}*- security against (non-uniform) poly*(κ)*-sized adversaries for an arbitrary small constant* $\epsilon > 0$*. Furthermore, for any* $x \in \{0,1\}^{2\kappa+\ell(\kappa)}$ *and distinct* $\mathsf{K},\mathsf{K}' \in \{0,1\}^{\eta(\kappa)}$*, we have that* $F_2(\mathsf{K},x) \neq F_2(\mathsf{K}',x)$.
- $\mathsf{G} : \{0,1\}^\kappa \to \{0,1\}^{2\kappa}$ *be a PRG which is* 2^{κ^ϵ}*- secure against (non-uniform) adversaries.*
- $\phi(C)$ *be the predicate asserting that* C *is a circuit of the form of Figure 4 with* F_1*,* F_2 *and* G *as specified above.*
- *Let* $c\mathcal{O} = (c\mathcal{O}.\mathsf{Obf}, c\mathcal{O}.\mathsf{Ver})$ *be a COA-secure obfuscator with respect to predicate* ϕ*.*

The encryption scheme $\mathcal{PKE} = (\mathsf{KeyGen},\mathsf{Enc},\mathsf{Dec})$ *where the three algorithms are as defined below:*

$\underline{\mathsf{KeyGen}(1^\kappa):}$
– *Sample keys* $\mathsf{K}_1, \mathsf{K}_2 \leftarrow \{0,1\}^{\eta(\kappa)}$.
– *Output* $\mathsf{pk} = \widehat{P}$, $\mathsf{sk} = (\mathsf{K}_1, \mathsf{K}_2)$,
where $\widehat{P} \leftarrow c\mathcal{O}.\mathsf{Obf}(1^\kappa, P_{\mathsf{K}_1,\mathsf{K}_2}, \phi)$, *and the program* $P_{\mathsf{K}_1,\mathsf{K}_2}$ *is defined in Figure 4.*

$\underline{\mathsf{Enc}(1^\kappa, \mathsf{pk}, m):}$
– *For* $i = 1, \ldots, \kappa$, *let* $\widetilde{P}_i \leftarrow_\$ c\mathcal{O}.\mathsf{Ver}(1^\kappa, \mathsf{pk}, \phi)$.
– *Sample* $r \leftarrow_\$ \{0,1\}^\kappa$, *let* $\sigma_i \leftarrow \widetilde{P}_i(m, r)$ *for* $i = 1 \ldots \kappa$.
– *If* $\sigma_1 = \cdots = \sigma_\kappa$, *then output* σ_1. *Else output* $\perp$.

$\underline{\mathsf{Dec}(\mathsf{sk} = (\mathsf{K}_1, \mathsf{K}_2), \sigma = (c_1, c_2, c_3)):}$
– *If* $c_3 \neq \mathsf{F}_2(\mathsf{K}_2, c_1 | c_2)$ *output* $\perp$.
Else, output $m' = \mathsf{F}_1(\mathsf{K}_1, c_1) \oplus c_2$

Hardwired: *Keys* $\mathsf{K}_1, \mathsf{K}_2 \in \{0,1\}^{\eta(\kappa)}$.
Input: *Message* $m \in \{0,1\}^{\ell(\kappa)}$, *random* $r \in \{0,1\}^\kappa$.

1. *Let* $c_1 = \mathsf{G}(r)$
2. *Let* $c_2 = F_1(\mathsf{K}_1, c_1) \oplus m$
3. *Let* $c_3 = F_2(\mathsf{K}_2, c_1 | c_2)$
4. *Output* $c = (c_1, c_2, c_3)$.

Figure 4: **Program** $P_{\mathsf{K}_1,\mathsf{K}_2}$.

Theorem 6.1. *Assume that* F_1, F_2, G *are subexponentially secure puncturable pseudorandom functions and pseudorandom generator as specified above, and that* $\mathcal{O}$ *is COA secure for circuits, with respect to the predicate* ϕ. *Then the above encryption scheme is Complete CCA secure as in Definition 6.1.*

Proof. Correctness is immediate. To see unique decryptability of $\mathcal{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, consider a putative public key $\mathsf{pk} \in \{0,1\}^{\mathsf{poly}(\kappa)}$, and let $\widetilde{P} \leftarrow_\$ c\mathcal{O}.\mathsf{Ver}(1^\kappa, \mathsf{pk}, \phi)$. From the soundness of $c\mathcal{O}$ we know that, barring negligible probability events, whenever $\widetilde{P} \neq \perp$ it holds that $\widetilde{P}$ is functionally equivalent to the program $P_{\mathsf{K}_1,\mathsf{K}_2}$ for some $\mathsf{K}_1, \mathsf{K}_2 \in \{0,1\}^{\eta(\kappa)}$. Let $k^* = (\mathsf{K}_1^*, \mathsf{K}_2^*)$ be the most probable value for these keys, that is $(\mathsf{K}_1^*, \mathsf{K}_2^*) = \arg\max_{(\mathsf{K}_1,\mathsf{K}_2)} \Pr[\widetilde{P} \leftarrow_\$ c\mathcal{O}.\mathsf{Ver}(1^\kappa, \mathsf{pk}, \phi) \& \widetilde{P} \equiv P_{\mathsf{K}_1,\mathsf{K}_2}]$.

Recall that Enc generates κ independent samples from $\widetilde{P}$ and outputs a ciphertext different than $\perp$ only if all the resulting ciphertexts are identical. Furthermore, the structure of $P_{\mathsf{K}_1,\mathsf{K}_2}$ guarantees that, whenever $\mathsf{K}_2 \neq \mathsf{K}_2'$, $P_{\mathsf{K}_1,\mathsf{K}_2}(m, r) \neq P_{\mathsf{K}_1',\mathsf{K}_2'}$ for all m, r. It follows that whenever Enc outputs a ciphertext different than $\perp$, all κ samples of $\widetilde{P}$ were functionally equivalent to the same $P_{\mathsf{K}_1,\mathsf{K}_2}$.

This means that if pk is not useless, then $\Pr[\widetilde{P} \leftarrow_\$ c\mathcal{O}.\mathsf{Ver}(1^\kappa, \mathsf{pk}, \phi) \ \& \ \widetilde{P} \equiv P_{\mathsf{K}_1^*,\mathsf{K}_2^*}] > 2/3$. It follows that $k^* = (\mathsf{K}_1^*, \mathsf{K}_2^*)$ is the opening of pk.

C-CCA security is shown by constructing, given an adversary $\mathcal{A}_{\mathsf{ccca}}$ that wins in the C-CCA game against the scheme with advantage ϵ, an admissible and ϕ-satisfying sampler Samp (see Definition 2.4), and an adversary $\mathcal{A}_{\mathsf{coa}}$ that wins with advantage ϵ the COA security game of Definition 6.1 with respect to Samp.

As a first step, it is instructive to consider the *symmetric* encryption scheme (E, D) that underlies $\mathcal{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$. That is, $E(k, m, r) = P_{\mathsf{K}_1,\mathsf{K}_2}(m, r)$, where $k = (\mathsf{K}_1, \mathsf{K}_2)$ and $P_{\mathsf{K}_1,\mathsf{K}_2}$ is presented in Figure 4. Similarly, $D(k, c) = \mathsf{Dec}(k, c)$. Observe that, as long as the PPRF families F_1, F_2 and the PRG G are subexponentially secure, (E, D) is a subexponentially secure CCA scheme. (Recall that CCA security for symmetric encryption provides the adversary with both encryption and decryption oracles.) Furthermore, it has the additional property that decrypting any ciphertext c with a random key results in $\perp$ except for subexponentially small probability. That is, $\Pr_k[\mathsf{Dec}(k, c) \neq \perp] < \mu(\kappa)$ for any c, where μ is subexponentially small.

The idea in constructing Samp is to have it generate triples of the form

$$(E(k_0, \cdot), E(k_1, \cdot), Z)$$

where k_0, k_1 are two randomly chosen encryption keys, and Z is a distribution over obfuscated programs whose code has both k_0, k_1, and: (a) given two messages $m_0, m_1 \in \{0,1\}^{\ell(\kappa)}$ and a valid ecryption under k_b, the program returns a challenge ciphertext c_b^* that encrypts m_b, and (b) the program implements encryption and decryption oracles under both k_0, k_1. (Jumping ahead, this will enable an adversary that

has a program $\zeta \leftarrow Z$ and oracle access to $E(k_b, \cdot)$ to essentially implement a CCA-attack on (E, D) with challenge ciphertext that's an encryption of m_b.)

We proceed to define Z more precisely. Let F be a subexponentially secure puncturable PRF whose range is the same as the domain of keys of (E, D), and consider the distribution over programs $W = \{W_{k_0,k_1,\rho,s,\tau}\}$, presented in Figure 5.

State: Encryption keys $k_\tau, k_{1-\tau}$, encryption randomness ρ, PRF key s.

Function:

- On input $(\text{'Enc'}, m, r)$, compute $r_\tau = F_s(\text{'Enc'}, m, r, \tau)$, $r_{1-\tau} = F_s(\text{'Enc'}, m, r, 1-\tau)$, $c_\tau = E(k_b, m, r_\tau)$, $c_{1-\tau} = E(k_b, m, r_{1-\tau})$. Return $c_\tau, c_{1-\tau}$.
- On input $(\text{'Chal'}, m_0, m_1, c)$, if there exists exactly one value $b \in \{0,1\}$ such that $D(k_b, c) \neq \perp$ then return $c^* = E(k_b, m_b, \rho)$. Else return $c^* = \perp$.
- On input $(\text{'Dec'}, c)$, if $c = c^*$ then return $\perp$. Else, if there exists exactly one value $b \in \{0,1\}$ such that $D(k_b, c) \neq \perp$ then return $D(k_b, c)$. Else return $\perp$.

Figure 5: Program $W_{k_0,k_1,\rho,s,\tau}$, where k_0, k_1 are encryption keys for (E, D), ρ is a random input for E, s is a key for F, and τ is a toggle bit. Note that τ does not affect the functionality of the program; it's purpose is to "randomize the code".

Now, let $\mathcal{O}$ be a piO obfuscator for circuits (as in Definition 2.5) and let $Z = \{Z_{k_0,k_1,\rho,s,\tau}\}$, where $Z_{k_0,k_1,\rho,s,\tau} = \mathcal{O}(W_{k_0,k_1,\rho,s,\tau})$ be the family of programs obtained by applying $\mathcal{O}$ to the programs in $W_{k_0,k_1,\rho,s\tau}(\cdot)$.

Claim 6.1. *The sampler* $\mathsf{Samp}(\kappa) \to (E(k_0, \cdot), E(k_1, \cdot), Z_{k_0,k_1,\rho,s,\tau})$ *is admissible and* ϕ*-satisfying.*

Proof. Samp is ϕ-satisfying by construction. Intuitively, admissibility follows from the fact that (E, D) is CCA secure, and so an adversary cannot tell whether a given ciphertext c^* is an encryption of m_0 or m_1, even when given access to a program (sampled from Z) that provides encryption and decryption service other than decrypting c^*.

More formally, the proof proceeds as follows: Let the family $W' = \{W'_{k_0,k_1,\rho,s,\tau}\}$ be identical to W, except that in response to input $(\text{'Chal'}, m_0, m_1, c)$, $W'_{k_0,k_1,\rho,s,\tau}$ computes $c_1 = E(k_1, m_0, r_1)$ (rather than $c_1 = E(k_1, m_1, r_1)$). That is, the challenge ciphertext c^* is always an encryption of m_0. Let $Z' = \{Z'_{k_0,k_1,\rho,s,\tau}\}$, where $Z'_{k_0,k_1,\rho,s,\tau} = \mathcal{O}(W'_{k_0,k_1,\rho,s,\tau})$.

We first observe that having oracle access to a program chosen from W is indistinguishable from having oracle access to a program chosen from W'. Furthermore, the distinguishing advantage is subexponentially small in the security parameter. In other words, the sampler $\mathsf{Samp}_0(\kappa) \to (W, W')$ is admissible. (This follows from the structure of the programs and the subexponential security of the PPRFs in use.)

Now, consider an adversary $\mathcal{A}$ that wins the admissibility game with respect to Samp with advantage ϵ. Using $\mathcal{A}$, we construct an adversary $\mathcal{A}'$ that distinguishes between Z and Z' with advantage ϵ. Since Samp_0 is admissible, adversary $\mathcal{A}'$ would contradict the premise that $\mathcal{O}$ is piO.

Adversary $\mathcal{A}'$ proceeds as follows. Given a program ζ (which is taken either from Z or from Z'), $\mathcal{A}'$ chooses $b \leftarrow \{0,1\}$ and runs $\mathcal{A}$ with input ζ, while emulating oracle access to $E(k_b, \cdot)$. That is, when $\mathcal{A}$ sends query (m, r) to its oracle, $\mathcal{A}'$ chooses a random r', obtains $c_0, c_1 \leftarrow \zeta(\text{'Enc'}, m, r')$, and returns c_b to $\mathcal{A}$. Finally, if $\mathcal{A}$ guesses b correctly, then $\mathcal{A}'$ decides that ζ is drawn from $Z_{k_0,k_1,\rho,s,\tau}$, else ζ is drawn from $Z'_{k_0,k_1,\rho,s,\tau}$.

We complete the proof of the claim by observing that if ζ is drawn from $Z_{k_0,k_1,\rho,s,\tau}$ then $\mathcal{A}$ sees an interaction that is subexponentially close to an interaction in the admissibility game for Samp, thus $\mathcal{A}$ predicts b with probability $1/2 + \epsilon$.[7] On the other hand, if ζ is drawn from $Z'_{k_0,k_1,\rho,s,\tau}$ then the view of $\mathcal{A}$ is statistically

[7] The subexponential difference results from the fact that in the latter case $\mathcal{A}$ sees an encryption oracle that responds to a query (m, r) with $E(k, m, r')$ where $r' = F_s(r)$, whereas in the former case the value r' is computed as $r' = F_s(r'')$, where r'' is an independently chosen random value.

independent from b. (Indeed, $\mathcal{A}$ is given oracle access to $E(k_b, \cdot)$ and auxiliary input drawn from $Z'_{k_0,k_1,\rho,s,\tau}$. But the distributions $k_0, W'_{k_0,k_1,\rho,s,\tau}$ and $k_1, W_{k_0,k_1,\rho,s,\tau}$ are identical: Fix some values for k_0, k_1, ρ, s; then the programs $W_{k_0,k_1,\rho,s,0}$ and $W_{k_1,k_0,\rho,s,1}$ are the same.) □

It remains to construct the COA adversary $\mathcal{A}_{\mathsf{coa}}$ given the putative C-CCA adversary $\mathcal{A}_{\mathsf{ccca}}$. Recall that $\mathcal{A}_{\mathsf{coa}}$ is given a program ζ sampled from $Z_{k_0,k_1,\rho,s,\tau}$, a program π^* which is sampled either from $c\mathcal{O}.\mathsf{Obf}(E(k_0,\cdot),\phi)$ or from $c\mathcal{O}.\mathsf{Obf}(E(k_1,\cdot),\phi)$, and access to deobfuscation oracle $\mathcal{D}$ that, given any valid program $\pi \neq \pi^*$, first samples $\tilde{\pi} \xleftarrow{\$} c\mathcal{O}.\mathsf{Ver}(\pi,\phi)$, and then returns the lexicographically first circuit C with $\phi(C) = 1$, that is functionally equivalent to $\tilde{\pi}$.

Adversary $\mathcal{A}_{\mathsf{coa}}$ runs $\mathcal{A}_{\mathsf{ccca}}$ with public encryption key $\mathsf{pk}^* = \pi^*$. When $\mathcal{A}_{\mathsf{ccca}}$ generates its encryption challenge m_0, m_1, $\mathcal{A}_{\mathsf{coa}}$ first obtains $c_0 = \pi^*(0, r)$ for a random r, and then computes $c^* = \zeta(\text{'Chal'}, m_0, m_1, c_0)$. Next, $\mathcal{A}_{\mathsf{coa}}$ returns c^* to $\mathcal{A}_{\mathsf{ccca}}$ as the challenge ciphertext. Decryption queries (pk, c) of $\mathcal{A}_{\mathsf{ccca}}$ are answered as follows:

1. If $\mathsf{pk} = \pi^*$ then respond with $\zeta(\text{'Dec'}, c)$.
2. If $\mathsf{pk} \neq \pi^*$, then $\mathcal{A}_{\mathsf{coa}}$ runs $\mathcal{D}(\mathsf{pk})$ for κ times. Let $C_1, ..., C_\kappa$ denote the resulting programs. For each C_i which is of the form P_{k_i}, let $m_i = \mathsf{Dec}(k_i, c)$. If there exists some $m_i \neq \perp$ then $\mathcal{A}_{\mathsf{coa}}$ returns the plurality value among the m_i's. Else $\mathcal{A}_{\mathsf{coa}}$ returns $\perp$.

Finally, $\mathcal{A}_{\mathsf{coa}}$ outputs the same bit as $\mathcal{A}_{\mathsf{ccca}}$.

We complete the proof by observing that, for $b = 0, 1$, when $\pi^* = c\mathcal{O}.\mathsf{Obf}(E(k_b,\cdot))$, the view of $\mathcal{A}_{\mathsf{ccca}}$ in the above execution is identical to its view in an actual C-CCA experiment, with a valid decryption oracle $\mathcal{D}$, and where the challenge ciphertext c^* is an encryption of m_b.

To see why the responses of $\mathcal{A}_{\mathsf{coa}}$ to the decryption queries of $\mathcal{A}_{\mathsf{ccca}}$ are identical to those of a valid C-CCA decryption oracle, consider a query (pk, c) made by $\mathcal{A}_{\mathsf{ccca}}$ where $\mathsf{pk} \neq \pi^*$ is not useless, and let $(\mathsf{K}_1^*, \mathsf{K}_2^*)$ be the opening of pk.

Recall that $\mathcal{A}_{\mathsf{coa}}$ obtains κ samples of $P_{\mathsf{K}_1,\mathsf{K}_2}$ that's functionally equivalent to $\widetilde{P} \leftarrow_{\$} c\mathcal{O}.\mathsf{Ver}(1^\kappa, \mathsf{pk}, \phi)$, and outputs the plurality value out of the κ decrypted values $\mathsf{Dec}((\mathsf{K}_1, \mathsf{K}_2), c)$. However, $\Pr_{\widetilde{P}\leftarrow_{\$} c\mathcal{O}.\mathsf{Ver}(1^\kappa,\mathsf{pk},\phi)}[\widetilde{P} \equiv P_{\mathsf{K}_1^*,\mathsf{K}_2^*}] \geq 2/3$, and for any $(\mathsf{K}_1', \mathsf{K}_2') \neq (\mathsf{K}_1^*, \mathsf{K}_2^*)$ we have $\Pr_{\widetilde{P}\leftarrow_{\$} c\mathcal{O}.\mathsf{Ver}(1^\kappa,\mathsf{pk},\phi)}[\widetilde{P} \equiv P_{\mathsf{K}_1',\mathsf{K}_2'}] < 1/3$. It follows that the plurality value differs from $\mathsf{Dec}((\mathsf{K}_1^*, \mathsf{K}_2^*), c)$ only with negligible probability. □

7 Structural Watermarking

In this section, we describe an application of COA-secure obfuscation to building watermarking schemes. As sketched and motivated in the Introduction, we present a new notion of watermarking, called *structural watermarking*, which is keyless, modifies the functionality of programs in a minimal way, and considers a broad class of counterfeiting attacks. We then show how COA-secure obfuscation can be used to construct structural watermarking schemes for a broad class of functionalities.

A structural watermarking scheme is specified with respect to an underlying class $\mathcal{C}$ of programs, the underlying family (or, distribution) of programs to be watermarked, and a "closeness relation" that determines the boundaries of "allowable similarity" between pairs of programs. More specifically:

Definition 7.1 (Structural Watermarking)**.** Let $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}}$ be a circuit class s.t. $\mathcal{C}_\kappa$ consists of circuits with input length $n(\kappa)$ and output length $m(\kappa)$. For a distribution family $\mathcal{D}_\mathcal{C} = \{\mathcal{D}_{\mathcal{C}_\kappa}\}_{\kappa\in\mathbb{N}}$ over $\mathcal{C}$ and a relation R over $\mathcal{C}$, a $(\mathcal{D}_\mathcal{C}, R)$*-structural watermarking scheme with a message space* $\mathcal{M} = \{\mathcal{M}_\kappa\}_{\kappa\in\mathbb{N}}$ consists of two PPT algorithms $(\mathsf{Mark}, \mathsf{Verify})$ as follows:

- $\mathsf{Mark}(1^\kappa, C, m)$: Mark is a randomized algorithm that takes as input a circuit $C \in \mathcal{C}_\kappa$, a message (or mark) $m \in \mathcal{M}_\kappa$ and outputs a (marked) circuit $\widehat{C}$.
- $\mathsf{Verify}(1^\kappa, \widehat{C})$: Verify is a randomized algorithm that takes as input a (purportedly marked) circuit $\widehat{C}$ and outputs a pair (C', m'), where C' is a circuit or $\perp$, and $m' \in \mathcal{M}_\kappa \cup \{\perp\}$.

The scheme is secure if there exists a negligible function ν s.t. the following properties hold:

- **Correctness.** For any circuit $C \in \mathcal{C}_\kappa$ and message $m \in \mathcal{M}_\kappa$ it holds that

$$\Pr_{(C',m') \leftarrow \mathsf{Verify}(1^\kappa, \mathsf{Mark}(1^\kappa, C, m))} [C' \not\equiv C \ \vee \ m' \neq m] \leq \nu(\kappa).$$

- $(\mathcal{D}_\mathcal{C}, R)$**-Unremovability.** For every non-uniform PPT adversary $\mathcal{A}$, for all sufficiently large κ,

$$\Pr[\mathsf{Exp}_{\mathcal{A}, \mathcal{D}_\mathcal{C}, R}(\kappa) = 1] \leq \nu(\kappa)$$

  where the experiment $\mathsf{Exp}_{\mathcal{A}, \mathcal{D}_\mathcal{C}, R}(\kappa)$ is defined as follows:

  1. $\mathcal{A}(1^\kappa)$ sends a message $m \in \mathcal{M}_\kappa$ to a challenger. The challenger samples a circuit $C \leftarrow \mathcal{D}_{\mathcal{C}_\kappa}$ and responds with $\widehat{C} \leftarrow \mathsf{Mark}(1^\kappa, C, m)$.
  2. $\mathcal{A}$ outputs a circuit C^*. Let $(\widetilde{C}^*, m^*) \leftarrow \mathsf{Verify}(1^\kappa, C^*)$. Then, the experiment outputs 1 iff $\widetilde{C}^* \neq \perp$, $m^* \neq m$, and
     - either $\exists C' \in \mathcal{C}_\kappa$ s.t. $C' \equiv \widetilde{C}^*$ and $R_\kappa(C', C) = 1$,
     - or there is no circuit in $\mathcal{C}_\kappa$ that is functionally equivalent to $\widetilde{C}^*$.

We note that, while the underlying family $\mathcal{C}$ of circuits may be naturally thought of representing the support of the distribution $\mathcal{D}$, it can also be thought of as significantly larger than the support of $\mathcal{D}$. This interpretation lends to situations where multiple instances of the watermarking scheme co-exist, using the same broad family $\mathcal{C}$, with different distributions, and different marks. It also means that for a specification $\mathcal{D}, R)$ to be realizable, the relation R has to respect $\mathcal{D}$, in the sense that it only accepts programs that are in the support of $\mathcal{D}$.

Our definition is incomparable with recent related definitions, specifically those of Cohen et al. [9] Aaronson et al. [1], where the latter proposes a unified definition to capture most prior works. Specifically, we require that a watermarking scheme has a verification algorithm that is executed before running the watermarked programs. In our definition, the adversary is considered to have removed the watermark only if it produces a circuit that verifies, and for which the corresponding circuit in the circuit family is related to the original circuit.

Our definition also strengthens the definitions from prior works (including [21] and [1]) in some crucial ways:

- Our definition eliminates the need for any key generation algorithm/public parameters.
- Our definition incorporates a guarantee that a circuit passing the verification indeed belongs to the circuit class.

In addition, our definition has a flavor of traitor-tracing security that is similar to the recent works of [15]. In particular, we say that an adversary wins the watermarking game if it removes/modifies the watermark and outputs a circuit that is *related* to the original circuit – where *related* refers to satisfying one of a large class of relations.

On the other hand, our definition is relaxed in that it only considers programs that are executed in a given execution environment (specifically, an environment where programs are derived via running an underlying verification algorithm). This is somewhat reminiscent of the relaxed notion in the recent work of Kitagawa et. al. [21] to capture publicly markable and extractable watermarking schemes without setup.

Construction. We construct a $(\mathcal{D}_\mathcal{C}, R)$-unremovable keyless verifiable watermarking scheme, when circuits drawn from $\mathcal{D}_\mathcal{C}$ are unlearnable from oracle access, but the relation R is such that a circuit becomes learnable given a related circuit (as made precise in Theorem 7.1). We first describe our construction before stating its security guarantee.

Construction 7.1. *Let $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}}$ be a circuit class s.t. $\mathcal{C}_\kappa$ consists of circuits that take inputs of length $n(\kappa)$ and produce outputs of length $m(\kappa)$, and $\mathcal{M} = \{\mathcal{M}_\kappa\}_{\kappa\in\mathbb{N}}$ be a space of polynomially long messages. For any $\kappa \in \mathbb{N}$, let $\mathcal{C}'_\kappa = \{C_m \mid C \in \mathcal{C}_\kappa, m \in \mathcal{M}_\kappa\}$, where*

$$C_m(x) = \begin{cases} m||C(0) & \text{if } x = 0 \\ C(x) & \text{otherwise.} \end{cases}$$

Let circuit class $\mathcal{C}' = \{\mathcal{C}'_\kappa\}_{\kappa\in\mathbb{N}}$ be the marked circuit class and ϕ' be its membership predicate, i.e. $\phi'(C) = 1$ iff $C \in \mathcal{C}'_\kappa$ (ϕ' will internally use $\phi_\mathcal{C}$, the membership predicate of $\mathcal{C}$).

Let $c\mathcal{O} = (c\mathcal{O}.\mathsf{Obf}, c\mathcal{O}.\mathsf{Ver})$ be COA-secure obfuscation for $\mathcal{C}'$, w.r.t. predicate ϕ' (according to Definition 3.3). Instantiate the watermarking scheme for $\mathcal{C}$ w.r.t. message space $\mathcal{M} = \{\mathcal{M}_\kappa\}_{\kappa\in\mathbb{N}}$ and relation R as follows:

- $\underline{\mathsf{Mark}(1^\kappa, C, m)}$*: Return $c\mathcal{O}.\mathsf{Obf}(1^\kappa, C_m, \phi')$, where C_m is defined using C as above.*
- $\underline{\mathsf{Verify}(1^\kappa, \widehat{C})}$*: Let $\widetilde{C}_{mark} \leftarrow c\mathcal{O}.\mathsf{Ver}(1^\kappa, \widehat{C}, \phi')$. Parse $\widetilde{C}_{mark}(0)$ as $m||y$, where $m \in \mathcal{M}_\kappa$ and $y \in \{0,1\}^{m(\kappa)}$. (If $\widetilde{C}_{mark} = \perp$, or the parsing above fails, return $(\perp, \perp)$). Construct a circuit $\widetilde{C}$ such that*

$$\widetilde{C}(x) = \begin{cases} y & \text{if } x = 0 \\ \widetilde{C}_{mark}(x) & \text{otherwise.} \end{cases}$$

Return $(\widetilde{C}, m)$.

We provide the following theorem which captures the security of the above construction.

Theorem 7.1. *Let $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}}$, $\mathcal{D}_\mathcal{C} = \{\mathcal{D}_{\mathcal{C}_\kappa}\}_{\kappa\in\mathbb{N}}$ and $R = \{R_\kappa\}_{\kappa\in\mathbb{N}}$ be ensembles of polynomial (in κ) sized circuits, distributions over those circuits and relations over those circuits, as follows:*

- *$\mathcal{C}_\kappa$ is a poly(κ)-time recognizable set of circuits taking $n(\kappa)$-bit inputs. Each circuit in $\mathcal{C}_\kappa$ has a unique $h(\kappa)$-bit encoding that is polynomial-time computable and invertible. Further, no two circuits in $\mathcal{C}_\kappa$ are functionally equivalent.*
- *(Unlearnability) For any circuit family $\mathcal{A} = \{\mathcal{A}_\kappa\}_{\kappa\in\mathbb{N}}$ where $\mathcal{A}_\kappa$ is of size $\mathsf{poly}(2^{n(\kappa)})$,*

$$\Pr_{C\leftarrow\mathcal{D}_{\mathcal{C}_\kappa}, C'\leftarrow\mathcal{A}_\kappa^{C(\cdot)}}[C' = C] \leq \mathsf{negl}(2^{n(\kappa)}).$$

- *(Reconstruction property) There is a family of polynomial (in κ) sized circuits $\mathrm{Rec} = \{\mathrm{Rec}_\kappa\}_{\kappa\in\mathbb{N}}$ such that,*

$$\Pr_{C\leftarrow\mathcal{D}_{\mathcal{C}_\kappa}}\left[\exists C' \in \mathcal{C}_\kappa,\ R_\kappa(C, C') = 1 \wedge \mathrm{Rec}_\kappa^{C(\cdot)}(C') \neq C\right] \leq \mathsf{negl}(\kappa).$$

Then the watermarking scheme in construction 7.1 is a $(\mathcal{D}_\mathcal{C}, R)$-unremovable keyless verifiable watermarking scheme, (according to Definition 7.1) for circuit class $\mathcal{C}$ and message space $\mathcal{M}$.

Proof. Let E_κ be a polynomial-sized circuit implementing a function $E_\kappa : \{0,1\}^{h(\kappa)} \times \{0,1\}^{n(\kappa)} \to \{0,1\}^{m(\kappa)}$, such that each circuit in $C \in \mathcal{C}_\kappa$ is equivalent to $E_\kappa(f, \cdot)$ for a unique (and efficiently computable) $f \in \{0,1\}^{h(\kappa)}$. (Concretely, we may consider the circuit E_κ such that $E_\kappa(f, x)$ first decodes f into a circuit C and then evaluates $C(x)$ using a universal circuit.)

To prove the theorem, we prove the given construction satisfies the two properties of Definition 7.1.

- **Correctness**: By perfect correctness of $c\mathcal{O}$ (Definition 3.3), we have

$$\Pr_{(C', m')\leftarrow\mathsf{Verify}(1^\kappa, \mathsf{Mark}(1^\kappa, C, m))}[C' \equiv C \wedge m' = m] = 1.$$

- **Unremovability**: Suppose for the sake of contradiction there exists a non-uniform PPT adversary $\mathcal{A}$ and a polynomial p, s.t. for infinitely many κ, $\Pr[\mathsf{Exp}_{\mathcal{A},\mathcal{D}_\mathcal{C},R}(\kappa) = 1] \geq 1/p(\kappa)$. We show a contradiction by using $\mathcal{A}$ to break the COA-security of $c\mathcal{O}$.

  Consider the following algorithms:

  - $\underline{\mathsf{Samp}(1^\kappa)}$:
    * $(m, \sigma) \leftarrow \mathcal{A}(1^\kappa)$, where $m \in \mathcal{M}_\kappa$ is the message sent by $\mathcal{A}$ and σ is the residual state of $\mathcal{A}$.
    * Sample $C^0, C^1 \leftarrow \mathcal{D}_{\mathcal{C}_\kappa}$ and compute their encodings $f^0, f^1 \in \{0,1\}^{h(\kappa)}$, so that $C^b(\cdot) \equiv E_\kappa(f^b, \cdot)$ for $b = 0, 1$. Sample $s \leftarrow \{0,1\}^{h(\kappa)}$ and set $z = (m, \sigma, s, \langle s, f^0 \rangle)$.
    * Let C^0_m, C^1_m be defined as follows: for $b \in \{0,1\}$,

$$C^b_m(x) = \begin{cases} m || C^b(0) & \text{if } x = 0 \\ C^b(x) & \text{otherwise.} \end{cases}$$

    * Output (C^0_m, C^1_m, z).
  - $\underline{\mathcal{D}^O(1^\kappa, \widehat{C}, z)}$:
    * Parse z as (m, σ, s, c). Initialize $\mathcal{A}$ with state σ and send $\widehat{C}$ to $\mathcal{A}$. Receive $\widehat{C}^*$ from $\mathcal{A}$.
    * Query O on $\widehat{C}^*$ to get C^*_{mark}. If $C^*_{mark} \neq \perp$, then parse $C^*_{mark}(0)$ as $m^* || y^*$ (where $m^* \in \mathcal{M}_\kappa$ and $y^* \in \{0,1\}^{m(\kappa)}$), and let C^* be defined as:

$$C^*(x) = \begin{cases} y^* & \text{if } x = 0 \\ C^*_{mark}(x) & \text{otherwise.} \end{cases}$$

    * Run $\widetilde{C}_{mark,orig} \leftarrow c\mathcal{O}.\mathsf{Ver}(1^\kappa, \widehat{C}, \phi')$. If $\widetilde{C}_{mark,orig} \neq \perp$, then parse $\widetilde{C}_{mark,orig}(0)$ as (m_{orig}, y_{orig}) and let $\widetilde{C}_{orig}$ be defined using y_{orig} and $\widetilde{C}_{mark,orig}$ as follows:

$$\widetilde{C}_{orig}(x) = \begin{cases} y_{orig} & \text{if } x = 0 \\ \widetilde{C}_{mark,orig}(x) & \text{otherwise.} \end{cases}$$

    * Run $C_{orig} \leftarrow \mathrm{Rec}_\kappa^{\widetilde{C}_{orig}(\cdot)}(C^*)$. Parse circuit C_{orig} as $E_\kappa(f_{orig}, \cdot)$ and check if $\langle s, f_{orig} \rangle = c$. If so, output 1. In all the other cases, return a uniform bit $\in \{0,1\}$.

Claim 7.1. Samp *is a ϕ'-satisfying admissible sampler (according to Definition 3.2), where ϕ' is the membership checking predicate for $\mathcal{C}'$ (defined using $\mathcal{C}$ in construction 7.1).*

Proof. On running Samp, we get (C^0_m, C^1_m, z). Since both $C^0_m, C^1_m \in \mathcal{C}'_\kappa$ and ϕ' is the membership checking predicate for $\mathcal{C}'$ we have that Samp is ϕ'-satisfying (according to Definition 3.1).

We next show that Samp is an admissible sampler (according to Definition 2.4). For $b \in \{0,1\}$, let $p_\mathcal{B}^{\mathsf{Samp},b,\kappa} := \Pr_{(C^0_m, C^1_m, z) \leftarrow \mathsf{Samp}(1^\kappa)}\left[\mathcal{B}^{C^b_m}(z) = 1\right]$ and let $\mathsf{Adv}_\mathcal{B}^{\mathsf{Samp},\kappa} := \left| p_\mathcal{B}^{\mathsf{Samp},0,\kappa} - p_\mathcal{B}^{\mathsf{Samp},1,\kappa} \right|$.

Consider the following sequence of experiments:

- Hybrid_0: Run $(C^0_m, C^1_m, z) \leftarrow \mathsf{Samp}(1^\kappa)$, where $z = (m, \sigma, s, \langle s, f^0 \rangle)$, and output $\mathcal{B}^{C^0_m}(z)$.
- Hybrid_1: Same as above, but after obtaining $(C^0_m, C^1_m, z) \leftarrow \mathsf{Samp}(1^\kappa)$, where $z = (m, \sigma, s, \langle s, f^0 \rangle)$, sample a uniform bit $c \leftarrow \{0,1\}$ and set $z' = (m, \sigma, s, c)$. Output $\mathcal{B}^{C^0_m}(z')$.
- Hybrid_2: Same as above, but output $\mathcal{B}^{C^1_m}(z')$.
- Hybrid_3: Same as above, but without replacing z by z', i.e. run $(C^0_m, C^1_m, z) \leftarrow \mathsf{Samp}(1^\kappa)$, and output $\mathcal{B}^{C^1_m}(z)$.

We have $\Pr[\mathsf{Hybrid}_0 = 1] = p_{\mathcal{B}}^{\mathsf{Samp},0,\kappa}$ and $\Pr[\mathsf{Hybrid}_3 = 1] = p_{\mathcal{B}}^{\mathsf{Samp},1,\kappa}$.

We show the following sub-claims:

SubClaim 7.1. $|\Pr[\mathsf{Hybrid}_0 = 1] - \Pr[\mathsf{Hybrid}_1 = 1]| \leq \mathsf{negl}(2^{n(\kappa)})$

Proof. Suppose for sake of contradiction, there exists some circuit family $\mathcal{B}$ of size $\mathsf{poly}(\kappa)$ and a polynomial p_2 s.t. for infinitely many κ, $|\Pr[\mathsf{Hybrid}_0 = 1] - \Pr[\mathsf{Hybrid}_1 = 1]| > 1/p_2(2^{n(\kappa)})$.

Note that between Hybrid_0 and Hybrid_1, $\mathcal{B}$ distinguishes $\langle s, f^0 \rangle$ from a uniform bit. We first build a corresponding guesser algorithm which predicts $\langle s, f^0 \rangle$ using $\mathcal{B}$ and then use the Goldreich-Levin decoder algorithm[8] to predict f^0 using just oracle access to the guesser algorithm, and which breaks the unlearnability property of $\mathcal{C}$, hence giving a contradiction.

W.l.o.g. assume that for a particular κ, $\Pr[\mathsf{Hybrid}_0 = 1] > \Pr[\mathsf{Hybrid}_1 = 1] + 1/p_2(2^{n(\kappa)})$ (the other case is handled similarly). Consider the guesser algorithm corresponding to $\mathcal{B}$, $\mathcal{G}^O(m, \sigma, s)$ (where O is a placeholder for any oracle $\mathcal{G}$ has access to), and which works as follows:

- Choose uniform $b \leftarrow \{0,1\}$.
- If $\mathcal{B}^O(m, \sigma, s, b) = 1$, then output b, else output $b \oplus 1$.

We then have the following sub-sub-claim that shows that $\mathcal{G}$ succeeds in predicting $\langle s, f^0 \rangle$ with good probability:

SubSubClaim 7.1.

$$\Pr\left[b' = \langle s, f^0 \rangle \ : \ \begin{array}{r} (C_m^0, C_m^1, z) \leftarrow \mathsf{Samp}(1^\kappa) \\ \textit{Parse } z = (m, \sigma, s, c) \\ b' \leftarrow \mathcal{G}^{C_m^0}(m, \sigma, s) \end{array}\right] \geq \frac{1}{2} + \frac{1}{p_2(2^{n(\kappa)})}$$

Proof. Opening up the working of $\mathcal{G}$ and conditioning on the bit chosen uniformly by it, the required probability above is equal to:

$$\begin{aligned}
&= \frac{1}{2}\Pr\left[b'' = 1 \ : \ \begin{array}{r} (C_m^0, C_m^1, z) \leftarrow \mathsf{Samp}(1^\kappa) \\ z = (m, \sigma, s, \langle s, f^0 \rangle), b \leftarrow \{0,1\} \\ b'' \leftarrow \mathcal{B}^{C_m^0}(m, \sigma, s, b) \end{array} \ \middle| \ b = \langle s, f^0 \rangle \right] \\
&+ \frac{1}{2}\Pr\left[b'' = 0 \ : \ \begin{array}{r} (C_m^0, C_m^1, z) \leftarrow \mathsf{Samp}(1^\kappa) \\ z = (m, \sigma, s, \langle s, f^0 \rangle), b \leftarrow \{0,1\} \\ b'' \leftarrow \mathcal{B}^{C_m^0}(m, \sigma, s, b) \end{array} \ \middle| \ b = \langle s, f^0 \rangle \oplus 1 \right] \\
&= \frac{1}{2}\Pr[\mathsf{Hybrid}_0 = 1] + \frac{1}{2}\Pr\left[b'' = 0 \ : \ \begin{array}{r} (C_m^0, C_m^1, z) \leftarrow \mathsf{Samp}(1^\kappa) \\ b'' \leftarrow \mathcal{B}^{C_m^0}(m, \sigma, s, \langle s, f^0 \rangle \oplus 1) \end{array}\right] \\
&= \frac{1}{2}\Pr[\mathsf{Hybrid}_0 = 1] + \frac{1}{2}p
\end{aligned}$$

where $p := \Pr\left[b'' = 0 \ : \ \begin{array}{r} (C_m^0, C_m^1, z) \leftarrow \mathsf{Samp}(1^\kappa) \\ b'' \leftarrow \mathcal{B}^{C_m^0}(m, \sigma, s, \langle s, f^0 \rangle \oplus 1) \end{array}\right]$. Now note that conditioning on the bit chosen uniformly in Hybrid_1, we get,

$$\begin{aligned}
\Pr[\mathsf{Hybrid}_1 = 1] &= \frac{1}{2}\Pr[\mathsf{Hybrid}_0 = 1] + \frac{1}{2}(1 - p) \\
\implies p &= \Pr[\mathsf{Hybrid}_0 = 1] + 1 - 2\Pr[\mathsf{Hybrid}_1 = 1]
\end{aligned}$$

[8] Recall that the Goldreich-Levin algorithm decodes $x \in \{0,1\}^n$, given access to any oracle O such that $\Pr_r[O(r) = \langle x, r \rangle] \geq \frac{1}{2} + \delta(n)$. This is done with success probability polynomial in $\delta(n)$ and with $poly(n, \log(\delta(n)^{-1}))$ queries [23].

Plugging the above back in, we get,

$$\begin{aligned}\text{Required probability } &= \frac{1}{2}\Pr[\mathsf{Hybrid}_0 = 1] + \frac{1}{2}(\Pr[\mathsf{Hybrid}_0 = 1] + 1 - 2\Pr[\mathsf{Hybrid}_1 = 1]) \\ &= \frac{1}{2} + (\Pr[\mathsf{Hybrid}_0 = 1] - \Pr[\mathsf{Hybrid}_1 = 1]) \\ &\geq \frac{1}{2} + \frac{1}{p_2(2^{n(\kappa)})}\end{aligned}$$

□

Now, given algorithms - $\mathcal{G}$ (the guesser algorithm corresponding to the distinguisher $\mathcal{B}$ between Hybrid_0 and Hybrid_1), and $\mathcal{A}$ (the original algorithm breaking the unremovability game), consider the following algorithm $\mathcal{A}_1^{C(\cdot)}(1^\kappa)$, which breaks the unlearnability property of $C \leftarrow \mathcal{D}_{\mathcal{C}_\kappa}$ as follows:

- $(m, \sigma) \leftarrow \mathcal{A}(1^\kappa)$.
- $\mathcal{A}_1$ reads the complete circuit C using oracle calls and locally defines $\widetilde{O}(s)$ as the algorithm which on input $s \in \{0,1\}^{h(\kappa)}$, runs $\mathcal{G}^{C_m}(m, \sigma, s)$, where C_m is the oracle defined as follows:

$$C_m(x) = \begin{cases} m||C(0) & \text{if } x = 0 \\ C(x) & \text{otherwise} \end{cases}$$

- Output $\mathsf{Dec}_{GL}^{\widetilde{O}(\cdot)}(1^{2^{n(\kappa)}})$, where Dec_{GL} is the Goldreich-Levin decoder algorithm.

We prove that $\mathcal{A}_1$ is of size $\mathsf{poly}(2^{n(\kappa)})$ and that there is a polynomial p_3 s.t.

$$\Pr_{C \leftarrow \mathcal{D}_{\mathcal{C}_\kappa}, C' \leftarrow \mathcal{A}_1^{C(\cdot)}(1^\kappa)} [\, C' \in \mathcal{C}_\kappa \wedge \forall x, C'(x) = C(x) \,] \geq 1/p_3(2^{n(\kappa)})$$

and hence breaking the unlearnability property. Note that the probability above involves the probability of sampling $(m, \sigma) \leftarrow \mathcal{A}(1^\kappa), C \leftarrow \mathcal{D}_{\mathcal{C}_\kappa}$. To use the Goldreich Levin decoder, we need that $\widetilde{O}(\cdot) \equiv \mathcal{G}^{C_m}(m, \sigma, \cdot)$ works only over the randomness of sampling its input $s \leftarrow \{0,1\}^{h(\kappa)}$. We thus first define a set $Good_\kappa$ which is the set of those (m, σ, C) for which only over the probability of sampling s, $\widetilde{O}(s)$ succeeds in finding $\langle s, f \rangle$ (where $C(\cdot) = E_\kappa(f, \cdot)$) with good probability. We then show (similarly to what's done in [23]) that over the randomness of sampling (m, σ) and C the set (m, σ, C) lies in the set $Good_\kappa$ with sufficiently larger than $1/2$ probability. Consider the following sub-subclaims:

SubSubClaim 7.2. *Let*

$$Good_\kappa = \left\{ (m, \sigma, C) \;\middle|\; \begin{array}{c} (m, \sigma) \in \mathsf{Supp}(\mathcal{A}(1^\kappa)), C \in \mathsf{Supp}(\mathcal{D}_{\mathcal{C}_\kappa}), C(\cdot) = E_\kappa(f, \cdot) \\ \Pr_{s \leftarrow \{0,1\}^{h(\kappa)}} \left[\mathcal{G}^{C_m}(m, \sigma, s) = \langle s, f \rangle \right] \geq \frac{1}{2} + \frac{1}{2p_2(2^{n(\kappa)})} \end{array} \right\}$$

Then,

$$\Pr_{(m,\sigma) \leftarrow \mathcal{A}(1^\kappa),\, C \leftarrow \mathcal{D}_{\mathcal{C}_\kappa}} [\, (m, \sigma, C) \in Good_\kappa \,] \geq \frac{1}{2p_2(2^{n(\kappa)})}$$

Proof. Opening the experiment for successful guessing of $\mathcal{G}$, we get using Subsubclaim 7.1,

$$\Pr\left[b = \langle s, f\rangle \;\middle|\; \begin{array}{c}(m,\sigma) \leftarrow \mathcal{A}(1^\kappa), C \leftarrow \mathcal{D}_{\mathcal{C}_\kappa}, s \leftarrow \{0,1\}^{h(\kappa)} \\ \text{Define } C_m \text{ using } (m,\ C), C(\cdot) = E_\kappa(f,\cdot) \\ b \leftarrow \mathcal{G}^{C_m}(m,\sigma,s)\end{array}\right] \geq \frac{1}{2} + \frac{1}{p_2(2^{n(\kappa)})}$$

$$\implies \Pr_{\substack{s\leftarrow\{0,1\}^{h(\kappa)} \\ b\leftarrow\mathcal{G}^{C_m}(m,\sigma,s)}}\left[b = \langle s,f\rangle \;\middle|\; (m,\sigma,C) \in Good_\kappa\right] \Pr_{\substack{(m,\sigma)\leftarrow\mathcal{A}(1^\kappa) \\ C\leftarrow\mathcal{D}_{\mathcal{C}_\kappa}}}[\,(m,\sigma,C) \in Good_\kappa\,]$$

$$+ \Pr_{\substack{s\leftarrow\{0,1\}^{h(\kappa)} \\ b\leftarrow\mathcal{G}^{C_m}(m,\sigma,s)}}\left[b = \langle s,f\rangle \;\middle|\; (m,\sigma,C) \notin Good_\kappa\right] \Pr_{\substack{(m,\sigma)\leftarrow\mathcal{A}(1^\kappa) \\ C\leftarrow\mathcal{D}_{\mathcal{C}_\kappa}}}[\,(m,\sigma,C) \notin Good_\kappa\,] \geq \frac{1}{2} + \frac{1}{p_2(2^{n(\kappa)})}$$

$$\implies 1 \times \left(\Pr_{m,\sigma,C}[\,(m,\sigma,C) \in Good_\kappa\,]\right) + \left(\frac{1}{2} + \frac{1}{2p_2(2^{n(\kappa)})}\right) \times 1 \geq \frac{1}{2} + \frac{1}{p_2(2^{n(\kappa)})}$$

$$\implies \Pr_{m,\sigma,C}[\,(m,\sigma,C) \in Good_\kappa\,] \geq \frac{1}{2p_2(2^{n(\kappa)})}$$

where in the second last implication we used that for $(m,\sigma,C) \notin Good_\kappa$, by definition of $Good_\kappa$, we have $\Pr_{\substack{s\leftarrow\{0,1\}^{h(\kappa)} \\ b\leftarrow\mathcal{G}^{C_m}(m,\sigma,s)}}[\,b = \langle s,f\rangle\,] < \frac{1}{2} + \frac{1}{2p_2(2^{n(\kappa)})}$. □

We then have the following:

SubSubClaim 7.3. *$\mathcal{A}_1$ is a $\mathsf{poly}(2^{n(\kappa)})$ size algorithm and there is a polynomial p_3 s.t.*

$$\Pr_{C\leftarrow\mathcal{D}_{\mathcal{C}_\kappa}, C'\leftarrow\mathcal{A}_1^{C(\cdot)}(1^\kappa)}[\,C' \in \mathcal{C}_\kappa \wedge \forall x, C'(x) = C(x)\,] \geq 1/p_3(2^{n(\kappa)})$$

Proof. From the definition of the set $Good_\kappa$, conditioned on any $(m,\sigma,C) \in Good_\kappa$, we know that $\Pr_s[\widetilde{O}(s) = \langle s,f\rangle] \geq \frac{1}{2} + \frac{1}{2p_2(2^{n(\kappa)})}$. Conditioned on the same, using the Goldreich-Levin decoder again (for strings of length $2^{n(\kappa)}$), we therefore have that there exists a polynomial p_4 s.t. Dec_{GL} is of size $\mathsf{poly}(2^{n(\kappa)})$ and

$$\Pr_{f'\leftarrow\mathsf{Dec}_{GL}^{\widetilde{O}(\cdot)}(1^{2^{n(\kappa)}})}[f' = C] \geq \frac{1}{p_4(2^{n(\kappa)})}.$$

Therefore, the probability in the given sub-sub-claim is:

$$\geq \Pr_{\widetilde{O}(\cdot)\equiv\mathcal{G}^{C_m}(m,\sigma,\cdot), f'\leftarrow\mathsf{Dec}_{GL}^{\widetilde{O}(\cdot)}(1^{2^{n(\kappa)}})}\left[\forall x, E_\kappa(f',x) = f(x) \;\middle|\; (m,\sigma,C) \in Good_\kappa\right] \cdot$$

$$\Pr_{m,\sigma,C}[\,(m,\sigma,C) \in Good_\kappa\,]$$

$$\geq \frac{1}{p_4(2^{n(\kappa)})} \times \frac{1}{2p_2(2^{n(\kappa)})} \leq \frac{1}{p_3(2^{n(\kappa)})}$$

for some polynomial p_3.

Finally, the size of $\mathcal{A}_1$ is bounded by the sum of (1) $2^{n(\kappa)}$ to read all values of f using oracle calls and (2) the size of $\mathsf{Dec}_{GL}^{\widetilde{O}(\cdot)}(1^{2^{n(\kappa)}})$. Recall that Dec_{GL} makes $\mathsf{poly}(2^{n(\kappa)})$ calls to $\widetilde{O}(\cdot)$, and each call to $\widetilde{O}$ involves running $\mathcal{G}^{C_m}(m,\sigma,\cdot)$ of size $\mathsf{poly}(\kappa)$. Hence, overall the size of $\mathcal{A}_1$ is bounded by $\mathsf{poly}(2^{n(\kappa)})$. □

□

SubClaim 7.2. $\Pr[\mathsf{Hybrid}_1 = 1] = \Pr[\mathsf{Hybrid}_2 = 1]$

Proof. The only difference between the two hybrids is that in Hybrid_1, $\mathcal{B}$ gets oracle access to C_m^0, while in Hybrid_2, it gets oracle access to C_m^1. Note that the input of $\mathcal{B}$ - z', is independent of either of C_m^0 or C_m^1. In addition, C_m^0, C_m^1 was defined identically using f^0, f^1 respectively, and both f^0, f^1 were in fact sampled identically from $\mathcal{D}_{\mathcal{C}_\kappa}$. Therefore, the two hybrids are the same experiment and the subclaim follows. □

SubClaim 7.3. $|\Pr[\mathsf{Hybrid}_2 = 1] - \Pr[\mathsf{Hybrid}_3 = 1]| \leq \mathsf{negl}(2^{n(\kappa)})$

Proof. This follows in the same way as the proof of SubClaim 7.1. □

Combining all the subclaims, we get

$$|\Pr[\mathsf{Hybrid}_0 = 1] - \Pr[\mathsf{Hybrid}_3 = 1]| \leq \mathsf{negl}(2^{n(\kappa)}) \leq \mathsf{negl}(\kappa)/2^{n(\kappa)}$$

This combined with the differing set $X = \{0,1\}^{n(\kappa)}$, implies that Samp is a ϕ'-satisfying admissible sampler. □

Continuing the proof, we now show that $\mathcal{D}$ together with Samp break the COA-security of $c\mathcal{O}$ (according to Definition 3.3). From the definition of COA security,

$$q_{c\mathcal{O},\mathcal{D}}^{\mathsf{Samp},b,\kappa} = \Pr_{\substack{(C_m^0, C_m^1, z) \leftarrow \mathsf{Samp}(1^\kappa) \\ \widehat{C} \leftarrow c\mathcal{O}.\mathsf{Obf}(1^\kappa, C_m^b, \phi')}} \left[\mathcal{D}^{\mathbb{O}(\kappa,\cdot) \circ c\mathcal{O}.\mathsf{Ver}(1^\kappa,\cdot,\phi') \circ \mathsf{Filt}_{\widehat{C}}}(\widehat{C}, z) = 1 \right].$$

Claim 7.2. $q_{c\mathcal{O},\mathcal{D}}^{\mathsf{Samp},1,\kappa} \geq \frac{1}{2} + \frac{1-\mathsf{negl}(\kappa)}{2p(\kappa)}$

Proof. Recall from how $\mathcal{D}$ is defined, $\mathcal{D}$ on input $\widehat{C}$ passes $\widehat{C}$ to $\mathcal{A}$ to get $\widehat{C}^*$. Note that the view of $\mathcal{A}$ is identical to $\mathsf{Exp}_{\mathcal{A},\mathcal{D}_\mathcal{C},R}(\kappa)$, and recall that by assumption for infinitely many κ, $\Pr[\mathsf{Exp}_{\mathcal{A},\mathcal{D}_\mathcal{C},R}(\kappa) = 1] \geq 1/p(\kappa)$. Suppose $\mathsf{Exp}_{\mathcal{A},\mathcal{D}_\mathcal{C},R}(\kappa) = 1$. This means $\mathcal{A}$ on input $\widehat{C}$ outputs $\widehat{C}^*$ s.t. if $(\widetilde{C}^*, m^*) \leftarrow \mathsf{Verify}(1^\kappa, \widehat{C}^*)$, then $\widetilde{C}^* \neq \bot, m^* \neq m$, and,

- either $\exists C^* \in \mathcal{C}_\kappa$ s.t. $C^* \equiv \widetilde{C}^*$ and $R_\kappa(C^*, C^1) = 1$,
- or there is no circuit in $\mathcal{C}_\kappa$ that is functionally equivalent to $\widetilde{C}^*$.

We first show that indeed there exists $C^* \in \mathcal{C}_\kappa$ s.t. $C^* \equiv \widetilde{C}^*$. Opening up $(\widetilde{C}^*, m^*) \leftarrow \mathsf{Verify}(1^\kappa, \widehat{C}^*)$ we have the following:

- $\widetilde{C}^*_{mark} \leftarrow c\mathcal{O}.\mathsf{Ver}(1^\kappa, \widehat{C}^*, \phi')$. If $\widetilde{C}^*_{mark} = \bot$, then output $(\bot, \bot)$.
- Else parse $\widetilde{C}^*_{mark}(0)$ as $m^*||y^*$, where $m^* \in \mathcal{M}_\kappa$ and $y^* \in \{0,1\}^{m(\kappa)}$. Construct circuit $\widetilde{C}^*$ as follows and return $(\widetilde{C}^*, m^*)$:

$$\widetilde{C}^*(x) = \begin{cases} y^* & \text{if } x = 0 \\ \widetilde{C}^*_{mark}(x) & \text{otherwise.} \end{cases}$$

Now since $\widetilde{C}^* \neq \bot$, this means $\widetilde{C}^*_{mark} \neq \bot$. Hence, by verifiability property of $c\mathcal{O}$ (definition 3.3),

$$\Pr_{\widetilde{C}^*_{mark} \leftarrow c\mathcal{O}.\mathsf{Ver}(1^\kappa, \widehat{C}^*, \phi')} \left[\exists C^*_{mark} \in \mathcal{C}'_\kappa : C^*_{mark} \equiv \widetilde{C}^*_{mark} \right] \geq 1 - \mathsf{negl}(\kappa).$$

By virtue of how $\mathcal{C}'_\kappa$ is defined, any $C^*_{mark} \in \mathcal{C}'_\kappa$ can be parsed as follows to get $C^* \in \mathcal{C}_\kappa$: parse $C^*_{mark}(0) = m^*||y^*$, where $m^* \in \mathcal{M}_\kappa, y^* \in \{0,1\}^{m(\kappa)}$, and let C^* be defined as:

$$C^*(x) = \begin{cases} y^* & \text{if } x = 0 \\ C^*_{mark}(x) & \text{otherwise.} \end{cases}$$

Since $\widetilde{C}^*$ is defined identically using $\widetilde{C}^*_{mark}$ as C^* is defined using C^*_{mark} and with overwhelming probability, $C^*_{mark} \equiv \widetilde{C}^*_{mark}$, we have that with overwhelming probability there exists $C^* \in \mathcal{C}_\kappa$ s.t. $C^* \equiv \widetilde{C}^*$. Suppose such a $C^* \in \mathcal{C}_\kappa$ does exist. Then, by the condition mentioned at the beginning of this proof, $R_\kappa(C^*, C^1) = 1$. Using the reconstruction property of $\mathcal{C}$ mentioned in the theorem statement, we have,

$$\Pr_{C^1 \leftarrow \mathcal{D}_{\mathcal{C}_\kappa}} \left[\exists C' \in \mathcal{C}_\kappa : R_\kappa(C', C^1) = 1 \wedge \mathrm{Rec}^{C^1(\cdot)}(C') \neq C^1 \right] \leq \mathsf{negl}(\kappa).$$

Hence with probability $(1-\mathsf{negl}(\kappa))$ over the sampling of C^1, since $R_\kappa(C^*, C^1) = 1$, we have $\mathrm{Rec}^{C^1(\cdot)}(C^*) = C^1$. Relating back to $\mathcal{D}$, if $\widetilde{C}_{mark,orig} \leftarrow c\mathcal{O}.\mathsf{Ver}(1^\kappa, \widehat{C}, \phi')$, by using perfect correctness of $c\mathcal{O}$, $\widetilde{C}_{mark,orig} \equiv C^1$, and therefore, $\mathrm{Rec}^{C^1(\cdot)}(C^*) = C^1$ is equivalent to $\mathrm{Rec}^{\widetilde{C}_{mark,orig}(\cdot)}(C^*) = C^1$.

Therefore, in this case, the check by $\mathcal{D}$ passes and it outputs 1. In all other cases it outputs a uniform bit, which implies the given claim.

□

Claim 7.3. $q^{\mathsf{Samp},0,\kappa}_{c\mathcal{O},\mathcal{D}} = \frac{1}{2}$

Proof. This follows in a similar way as above, except that we get $\mathrm{Rec}^{\widetilde{C}_{mark,orig}(\cdot)}(C^*) = C^0$ and the check $\langle s, f^0 \rangle = c$ passes with probability $1/2$, hence, making $\mathcal{D}$ always output a uniform bit. Note that the above holds because $\langle s, f^0 \rangle = c$ is equivalent to $\langle s, f^0 \rangle = \langle s, f^1 \rangle$, which happens with only probability $1/2$ since $s \in \{0,1\}^{h(\kappa)}$ was sampled uniformly. □

Therefore, combining all our claims, Samp is a ϕ'-satisfying admissible sampler, while $\mathsf{COAAdv}^{\mathsf{Samp},\kappa}_{c\mathcal{O},\mathcal{D}} = \left| q^{\mathsf{Samp},0,\kappa}_{c\mathcal{O},\mathcal{D}} - q^{\mathsf{Samp},1,\kappa}_{c\mathcal{O},\mathcal{D}} \right| \geq \frac{1-\mathsf{negl}(\kappa)}{2p(\kappa)}$, implying a contradiction to the COA-security of $c\mathcal{O}$, and hence, the unremovability claim follows.

□

Next, we provide the following corollary which captures PRF watermarking as special case of the above theorem.

Corollary 7.1. *Let $F = \{F_k(\cdot)\}_{k \in \mathcal{K}_\kappa, \kappa \in \mathbb{N}}$ be a PRF family with key-space $\mathcal{K} = \{\mathcal{K}_\kappa\}_{\kappa \in \mathbb{N}}$, and seed, input, and output lengths as polynomials $h(\kappa), n(\kappa)$ and $m(\kappa)$ respectively, such that $n(\kappa) \leq \kappa^c$ for some $c < 1$. In addition, suppose the key distribution ensemble $\mathcal{D}_\mathcal{K}$ and relation ensemble R are as follows:*

- *F is a sub-exponentially secure PRF under key distribution $\mathcal{D}_\mathcal{K}$. That is, for any adversary of size $\mathsf{poly}(2^{n(\kappa)})$, the following holds: (where $\mathcal{F}(n,m)$ = set of all functions with input length n and output length m)*

$$\left| \Pr_{k \leftarrow \mathcal{D}_\mathcal{K},\, b \leftarrow \mathcal{A}^{F_k(\cdot)}(1^\kappa)} [b = 1] - \Pr_{H \leftarrow \mathcal{F}(n,m),\, b \leftarrow \mathcal{A}^{H}(1^\kappa)} [b = 1] \right| \leq \mathsf{negl}(2^{n(\kappa)})$$

- *There exists an algorithm* Rec *s.t.*

$$\Pr_{k \leftarrow \mathcal{D}_\mathcal{K}} \left[\exists k' \in \mathcal{K},\, R_\kappa(k', k) = 1 \wedge \mathrm{Rec}^{F_k(\cdot)}_\kappa(k') \neq k \right] = \mathsf{negl}(\kappa).$$

Then the watermarking scheme for F in construction 7.1 is a $(\mathcal{D}_{\mathcal{K}}, R)$-unremovable keyless verifiable watermarking scheme.

As a concrete instantiation of the above corollary, we consider the following relation over PRF keys: $R_\kappa(k, k') = 1$ iff $F_k(\cdot)$ agrees with $F_{k'}(\cdot)$ on at least one input. We will use a sub-exponentially secure PRF family F, which satisfies the following *key injectivity* property:

$$\Pr_{k \leftarrow \mathcal{D}_{\mathcal{K}_\kappa}} [\exists k' \in \mathcal{K},\ R_\kappa(k, k') = 1 \wedge k' \neq k] = \mathsf{negl}(\kappa).$$

where $\mathcal{D}_{\mathcal{K}_\kappa}$ denotes the key distribution for which the PRF security holds. Such PRFs can be constructed as in [9] under sub-exponential DDH and LWE assumptions. For such a PRF, $R(k, k') = 1$ iff $k = k'$ (for most k). Then, letting Rec be the identity function satisfies the condition on the relation R in the above corollary. Thus, instantiating Corollary 7.1 with $F_k(\cdot), \mathcal{D}_{\mathcal{K}}, R$ as defined above, we get a $(\mathcal{D}_{\mathcal{K}}, R)$-keyless verifiable watermarking scheme for F.

8 Constructing Verifiability Fortifiers

This section demonstrates that the construction from Section **??** continues to provide meaningful security even when instantiated with non-robust primitives (which may be more efficient or obtainable from weaker assumptions). Specifically, that construction provides plain verifiability fortification as long as the NIDI is secure and the underlying commitment is statistically binding:

Theorem 8.1. *Assuming the existence of NIDI arguments satisfying $\epsilon(\kappa)$-gap distributional indistinguishability according to Definition 2.2, there exists an $\epsilon(\kappa)$-gap verifiability fortification for obfuscation satisfying Definition 3.4.*

Construction 8.1. *Define $\mathcal{L}_\phi = \left\{\widetilde{C}_\kappa : \exists (C_\kappa, r) \text{ such that } \widetilde{C}_\kappa = \mathcal{O}(C_\kappa; r) \text{ and } \phi(C_\kappa) = 1\right\}_{\kappa \in \mathbb{N}}$.*

1. *$v\mathcal{O}.\mathsf{Obf}$: The obfuscate algorithm $v\mathcal{O}.\mathsf{Obf}(1^\kappa, C)$ does the following:*
   - *Define distribution $\mathcal{D}_C(r) = \mathcal{O}(C; r)$ for uniformly sampled r.*
   - *Output $\pi = \mathsf{NIDI}.\mathcal{P}(1^\kappa)$ for the language $\mathcal{L}_\phi$ computed using uniform randomness $r_{\mathcal{C}}$.*
2. *$v\mathcal{O}.\mathsf{Verify}$: The obfuscate algorithm $v\mathcal{O}.\mathsf{Verify}(1^\kappa, \pi, \mathcal{L}_\phi)$ does the following:*
   - *Sample randomness $r_{\mathcal{R}}$.*
   - *Obtain $y \leftarrow \mathsf{NIDI}.\mathcal{V}(1^\kappa, \pi; r_{\mathcal{R}})$ for the language $\mathcal{L}_\phi$.*
   - *Output y.*

Proof of Security. In short, completeness, verifiability, and ϵ-gap indistinguishability of obfuscated circuits according to Definition 3.4 follows from the corresponding properties of NIDI (Definition 2.2) and correctness of underlying obfuscation scheme. Below we provide a more formal treatment.

Completeness. Let $y = v\mathcal{O}.\mathsf{Verify}(v\mathcal{O}.\mathsf{Obf}(1^\kappa, C)) = \mathsf{NIDI}.\mathcal{V}(1^\kappa, \mathsf{NIDI}.\mathcal{P}(1^\kappa, \mathcal{D}_C); r_{\mathcal{R}})$. By completeness of NIDI, it immediately follows that $y \in \mathsf{Supp}(\mathcal{D}_C)$, i.e. there exists some r' such that $y = \mathcal{O}(C; r')$. By perfect correctness of $\mathcal{O}$, it follows that y and C are functionally equivalent; thus completeness holds.

Verifiability. Since $v\mathcal{O}.\mathsf{Verify}(1^\kappa, \pi) = \mathsf{NIDI}.\mathcal{V}(1^\kappa, \pi; r_{\mathcal{R}})$ for uniform $r_{\mathcal{R}}$, and since, by soundness of NIDI, for every ensemble of polynomial-length strings $\{\Pi_\kappa\}_\kappa$ there exists a negligible function μ such that

$$\Pr_{x \leftarrow \mathsf{NIDI}.\mathcal{V}(1^\kappa, \Pi_\kappa, \mathcal{L}_\phi)} \left[(x \neq \bot) \wedge (x \notin \mathcal{L}_\phi)\right] \leq \mu(\kappa),$$

Thus, the verifiability of $v\mathcal{O}$ follows immediately.

ϵ-Gap Indistinguishability of Obfuscated Circuits. Let T be a transformation on distinguishers guaranteed to exist by ϵ-gap distributional indistinguishability of the NIDI scheme. We claim that this T also satisfies the requirement of ϵ-gap indistinguishability of obfuscated circuits. Indeed, for any PPT distinguisher D,

$$\mathsf{Adv}^{\mathsf{Samp}}_{v\mathcal{O}.\mathsf{Obf},\mathcal{D}} = \Big|\Pr[D(\mathcal{P}(1^\kappa,\mathcal{D}_0))=1] - \Pr[D(\mathcal{P}(1^\kappa,\mathcal{D}_1))=1]\Big|$$

$$\leq \frac{1}{\epsilon(\kappa)}\Big|\Pr[T(D)(\mathcal{X}_0)=1] - \Pr[T(D)(\mathcal{X}_1)=1]\Big| = \frac{1}{\epsilon(\kappa)}\mathsf{Adv}^{\mathsf{Samp}}_{\mathcal{O},T(\mathcal{D})},$$

and therefore

$$\mathsf{Adv}^{\mathsf{Samp}}_{\mathcal{O},T(\mathcal{D})} \geq \epsilon(\kappa)\cdot\mathsf{Adv}^{\mathsf{Samp}}_{v\mathcal{O}.\mathsf{Obf},\mathcal{D}}.$$

References

[1] Scott Aaronson, Jiahui Liu, Qipeng Liu, Mark Zhandry, and Ruizhe Zhang. New approaches for quantum copy-protection. In *Annual International Cryptology Conference*, pages 526–555. Springer, 2021.

[2] Saikrishna Badrinarayanan, Vipul Goyal, Aayush Jain, and Amit Sahai. Verifiable functional encryption. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016*, volume 10032 of *Lecture Notes in Computer Science*, pages 557–587, 2016. Full version at https://eprint.iacr.org/2016/629.pdf.

[3] Boaz Barak. How to go beyond the black-box simulation barrier. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 106–115, 2001.

[4] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6:1–6:48, 2012.

[5] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Candidate io from homomorphic encryption schemes. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 79–109. Springer, 2020.

[6] Ran Canetti, Huijia Lin, and Rafael Pass. Adaptive Hardness and Composable Security in the Plain Model from Standard Assumptions. In *FOCS 2010*, pages 541–550, 2010.

[7] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 468–497. Springer, 2015.

[8] Ran Canetti and Mayank Varia. Non-malleable obfuscation. In Omer Reingold, editor, *TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, volume 5444 of *Lecture Notes in Computer Science*, pages 73–90. Springer, 2009.

[9] Aloni Cohen, Justin Holmgren, Ryo Nishimaki, Vinod Vaikuntanathan, and Daniel Wichs. Watermarking cryptographic capabilities. *SIAM J. Comput.*, 47(6):2157–2202, 2018.

[10] Lalita Devadas, Willy Quach, Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Succinct LWE sampling, random polynomials, and obfuscation. In Kobbi Nissim and Brent Waters, editors, *Theory of Cryptography - 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part II*, volume 13043 of *Lecture Notes in Computer Science*, pages 256–287. Springer, 2021.

[11] Marc Fischlin. Completely non-malleable schemes. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 779–790. Springer, 2005.

[12] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 467–476, 2013.

[13] Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 736–749. ACM, 2021.

[14] Rishab Goyal, Sam Kim, Nathan Manohar, Brent Waters, and David J. Wu. Watermarking public-key cryptographic primitives. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 367–398. Springer, 2019.

[15] Rishab Goyal, Sam Kim, Brent Waters, and David J Wu. Beyond software watermarking: Traitor-tracing for pseudorandom functions. *IACR Cryptol. ePrint Arch.*, 2020:316, 2020.

[16] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. Cryptology ePrint Archive, Report 2020/1003, 2020. https://eprint.iacr.org/2020/1003.

[17] Yael Tauman Kalai and Dakshita Khurana. Non-interactive non-malleability from quantum supremacy. In *CRYPTO 2019*, pages 552–582, 2019.

[18] Dakshita Khurana. Non-interactive distributional indistinguishability (NIDI) and non-malleable commitments. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part III*, volume 12698 of *Lecture Notes in Computer Science*, pages 186–215. Springer, 2021.

[19] Sam Kim and David J. Wu. Watermarking cryptographic functionalities from standard lattice assumptions. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 503–536. Springer, 2017.

[20] Sam Kim and David J. Wu. Watermarking cryptographic functionalities from standard lattice assumptions. *J. Cryptol.*, 34(3):28, 2021.

[21] Fuyuki Kitagawa, Ryo Nishimaki, and Takashi Yamakawa. Secure software leasing from standard assumptions. *arXiv preprint arXiv:2010.11186*, 2020.

[22] Ilan Komargodski and Eylon Yogev. Another step towards realizing random oracles: Non-malleable point obfuscation. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018*, volume 10820 of *Lecture Notes in Computer Science*, pages 259–279. Springer, 2018.

[23] Leonid A. Levin. One-way functions and pseudorandom generators. *Combinatorica*, 7(4):357–363, 1987.

[24] Huijia Lin, Rafael Pass, and Pratik Soni. Two-round and non-interactive concurrent non-malleable commitments from time-lock puzzles. In Chris Umans, editor, *FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 576–587. IEEE Computer Society, 2017.

[25] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC 1990*, page 427–437, New York, NY, USA, 1990. Association for Computing Machinery.

[26] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer, 1991.

[27] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 475–484. ACM, 2014.

[28] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. *SIAM Journal on Computing*, 50(3):857–908, 2021.

[29] Carmine Ventre and Ivan Visconti. Completely non-malleable encryption revisited. In Ronald Cramer, editor, *PKC 2008*, volume 4939 of *Lecture Notes in Computer Science*, pages 65–84. Springer, 2008.